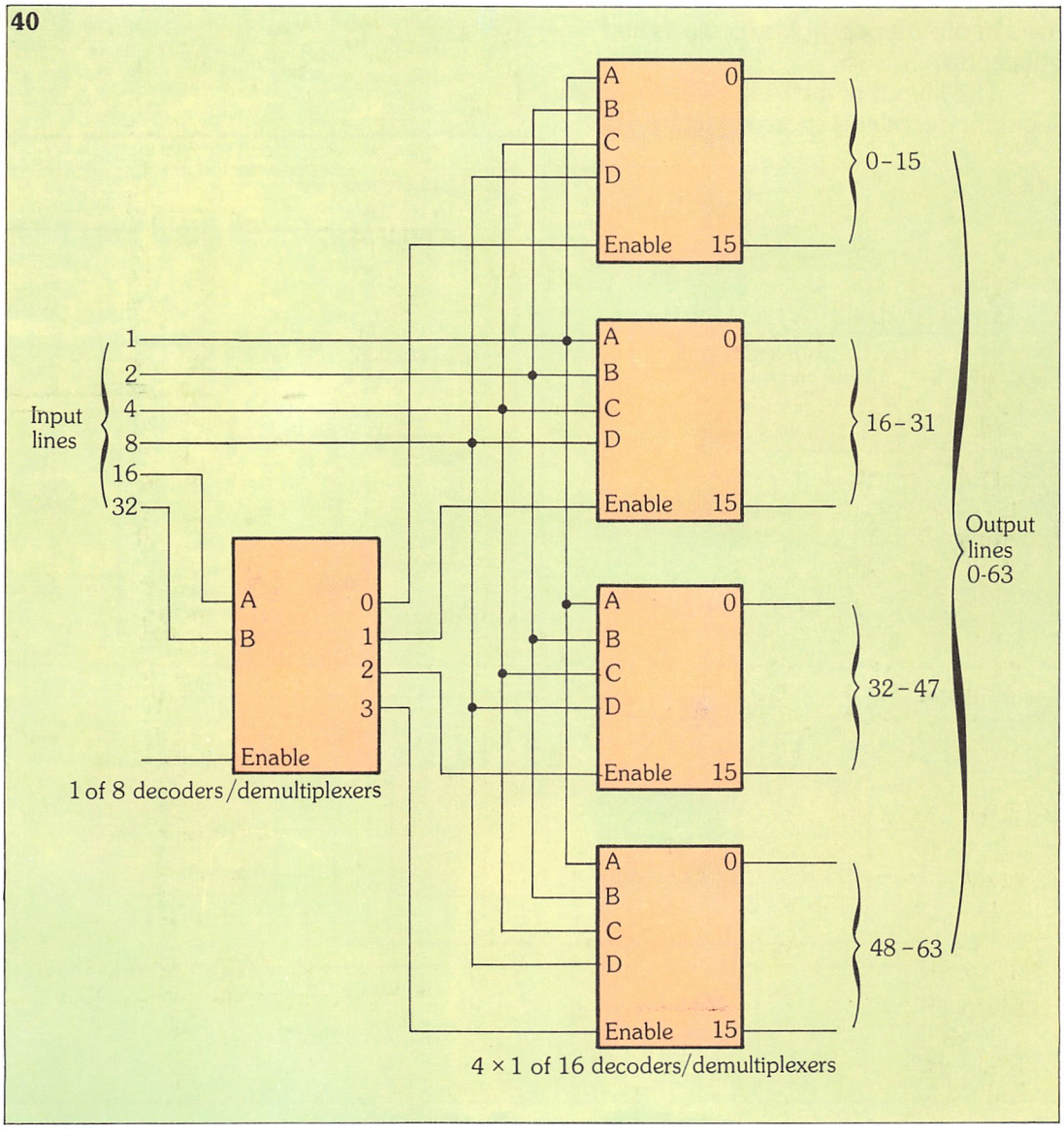
(continued from part 9)

The 74155/6 circuits can also be used as three to eight line decoders, one to eight line demultiplexers and dual one to four line demultiplexers. To change their use in this manner the enabling inputs have to be connected in different ways.

40. Schematic diagram illustrating how 6 input lines can be decoded into one of sixty four output lines.



Decoder networks

The circuit in *figure 40* shows how six input lines can be decoded into one of sixty-four output lines. Decoding network circuits with virtually any number of input lines can be designed in this way by cascading decoders; their correct functioning depends upon the enabling signals sent by the first decoder circuit.

Four line to seven line decoder

Seven-segment displays are the devices most commonly used for the number displays of calculators and other digital equipment. Coded numerical information needs to be converted into signals which light up the relevant segments. The code conversion is done by four line to seven line decoder/drivers, also known as BCD-to-seven-segment decoder/drivers.

A seven-segment display gives a total of $2^7 = 128$ possible combinations. The most important are those which make up the numerals 0 to 9. *Figure 41* is a truth table for a seven-segment display, showing a lighted segment by a logic 0 and unlit segments by logic 1. Note that a BCD-to-

seven line decoder does not give a '1 of 7' output, but anything from two to seven outputs can be on at one time.

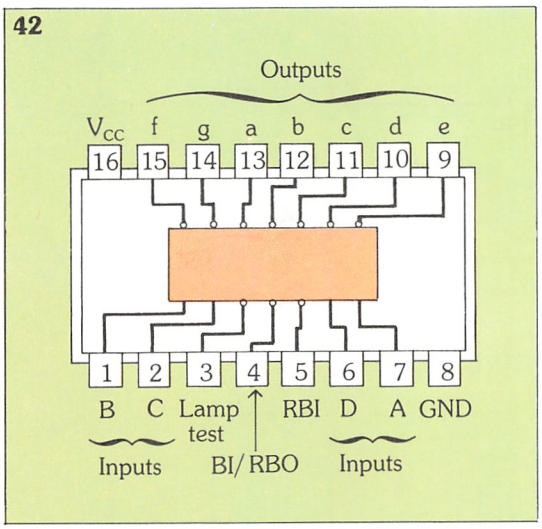
The seven-segment displays used in modern calculators are operated by decoder/drivers which are an integral part of the main calculator chip. However, it is useful to examine the way in which the simpler circuits operate, as they are still widely used in prototypes, hobby projects and education.

The layout of the 7447 BCD-to-seven segment decoder is shown in *figure 42*.

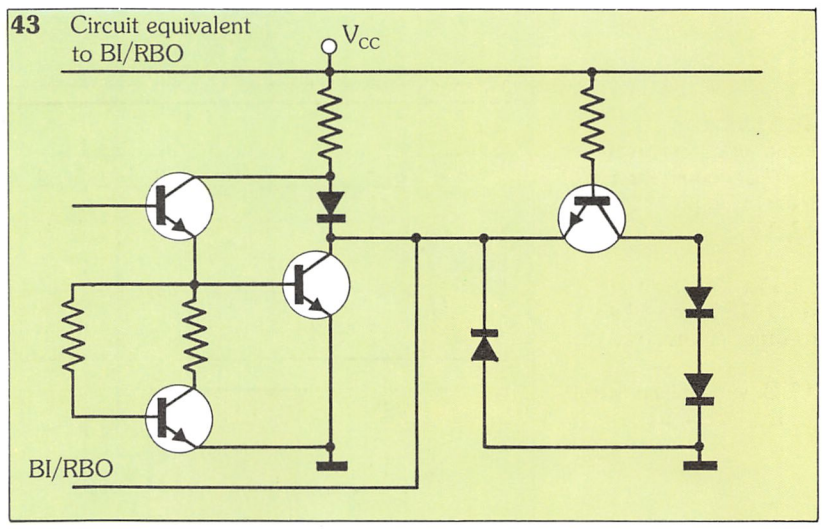
41

Decimal number	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	1	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0

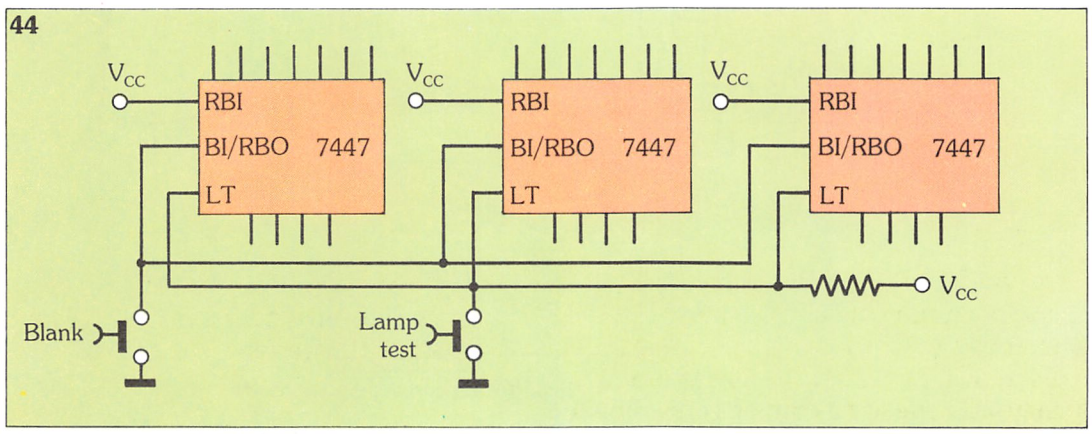
42



43 Circuit equivalent to BI/RBO



44



41. Truth table for a BCD-to-seven-segment decoder.

42. Pin configuration for the 7447 BCD-to-seven-segment decoder.

43. I/O configuration for the BI/RBO circuit.

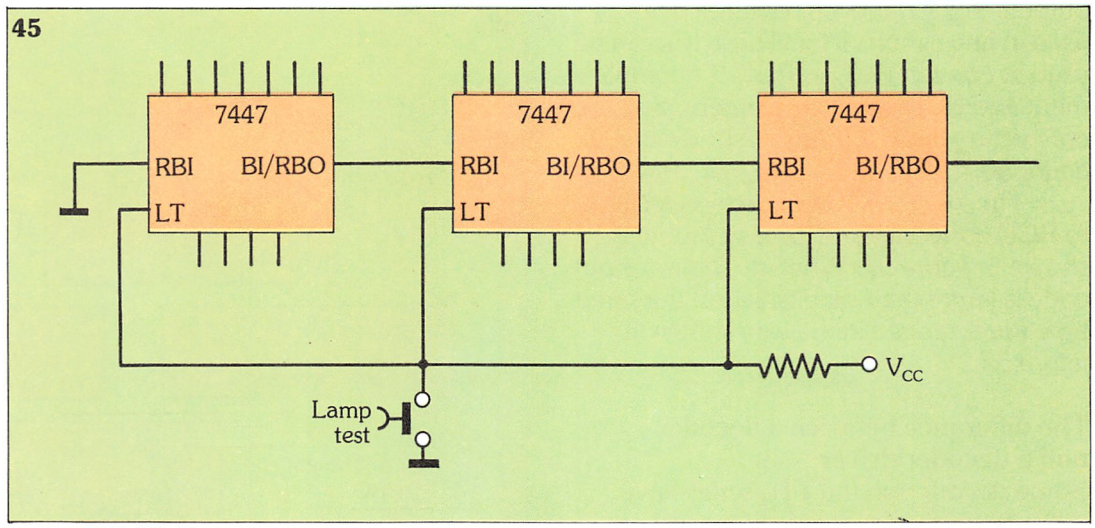
44. Use of the blank pin to switch the display off.

Pins 3, 4 and 5 are the connections for the lamp test input, blanking input/ripple blanking output, and ripple blanking input.

By setting the lamp test (**LT**) terminal to logic 0, all the display segments will light up, enabling the correct functioning of the display to be tested. Setting the blanking input/ripple blanking output (**BI/RBO**) pin to logic 0 will give the blank function and all the segments in the display will be turned off. The diagram of the BI/RBO circuit is shown in *figure 43*.

If BI/RBO, in *figure 42*, is disconnected from logic 0, inputs DCBA and LT are set to logic 0, then a logic 0 at **RBI** will produce a blank on the display and a 0 on BI/RBO (which now acts as an output). This is a very useful function as it can be used to turn *off* all the zeros to the left of the most significant digit of a displayed

45. Cascade connection of three 7447 decoders (each with its own driver) using the blank input.

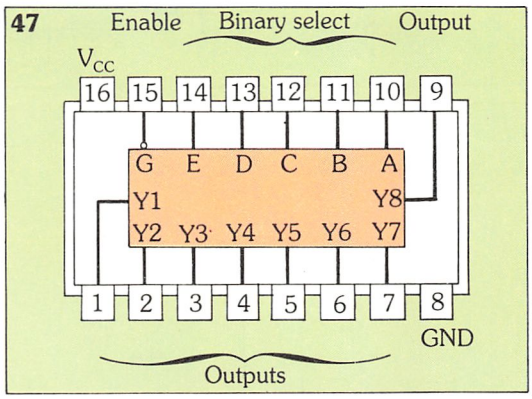


46. Output configurations for the 7447 decoder when given the inputs 10, 11, 12, 13, 14 and 15.

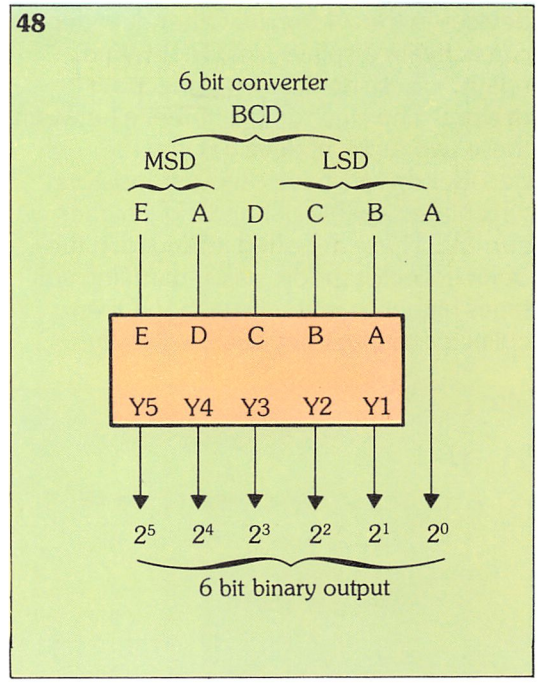
46

D	C	B	A	a	b	c	d	e	f	g
1	0	1	0	1	1	1	0	0	1	0
1	0	1	1	1	1	0	0	1	1	0
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	0	1	1	0	1	0	0
1	1	1	0	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1

47. Pin configuration for the 74184 6-bit BCD-to-binary converter.



48. Basic configuration for the 74184 IC.



number. This is illustrated in *figure 44*. Remember, using a basic IC like the 7447, each seven-segment display will have to have its own driver, and to operate correctly they will have to be connected as shown in *figure 45*.

If LT and RBI are set to logic 1 and BI/RBO is disconnected from logic 0 then the truth table in *figure 41* is valid for the 7447 chip. On the other hand, if positive logic is being used, then the 7448 decoder/driver has to be employed. In this case, an output of logic 1 would refer to a lighted segment and 0 to an unlit one.

If the 7447 is given the inputs, 10, 11, 12, 13, 14 and 15 it produces the outputs shown in *figure 46*. These do not correspond to the decimal numbers 10 to 15 as this IC cannot supply outputs for two decimal digits.

BCD to binary and binary to BCD converters

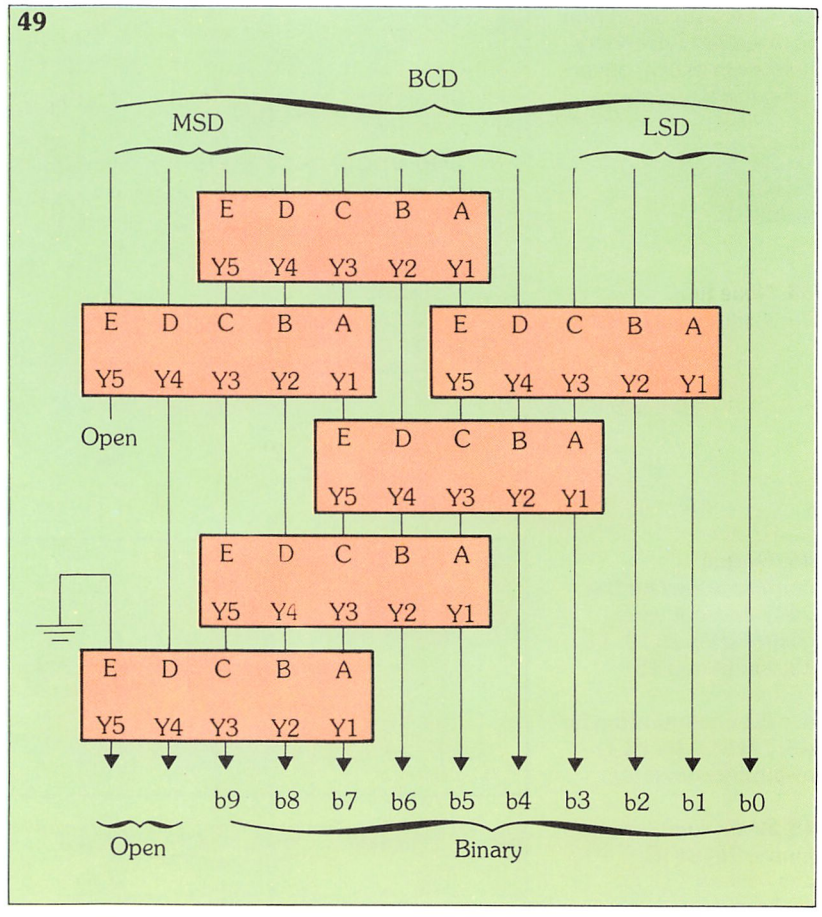
The 74184 IC is a 6-bit BCD to binary converter and is shown in *figure 47*. Its basic connections are shown in *figure 48*. The number of ICs used increases rapidly with the number of BCD digits to be converted, as shown in *figure 49*. To take this further, the conversion of six BCD decades into binary would need 28 separate ICs. You may have noticed that the

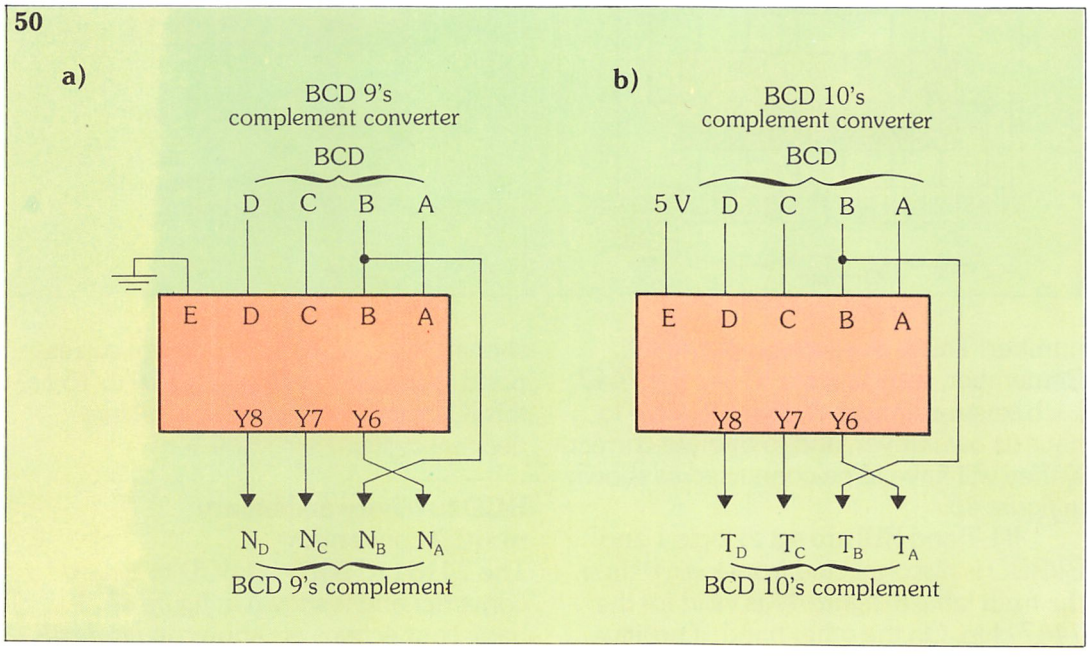
outputs Y6, Y7 and Y8 have not been used in any circuit so far. These ICs can be used to convert BCD to BCD 9's complement and BCD 10's complement, by connecting these outputs as shown in *figure 50*.

The 74185 A IC only acts as a binary to BCD converter and its basic layout is shown in *figure 51*. When the binary input code is larger than six bits, ICs of the same type are arranged in the way shown in *figure 52*.

The difference between a decoder and a decoder/driver

Some decoders in the TTL series have outputs which can only feed into other devices in the 74 family. Other decoders can activate displays, small lights and relays – these are known as **decoder/drivers**. The electronic difference between these two types of device is that the decoder drivers have output transistors which are capable of handling a higher current. These are often arranged in the open collector mode, easily handling voltages higher than the usual 5 V. Open collector outputs can also be directly





49. Converters connected in cascade. The greater the number of BCD digits to be converted; the greater the number of ICs used.

50. Converting BCD to: **(a)** BCD 9's complement and **(b)** BCD 10's complement.

51. Basic configuration of the 74185A binary-to-BCD converter.

52. 74185As connected in cascade: (a) with 8-bit binary input; **(b)** with 12-bit binary input.

connected together.

To summarize: a decoder circuit can do the following:

- select or enable a particular device in the same logic family;
- address a binary signal along one out of many lines. This is a demultiplexer function and will be discussed later;
- simply convert one digital code into another, more suitable for the receiving

circuits.

A decoder/driver can do all of the above as well as drive devices such as numerical displays, small incandescent light bulbs etc.

Some decoder/drivers are available for specialised applications, a good example is the 74141 IC (*figure 53*), which is designed to drive a Nixie tube. A Nixie tube is a numerical display device, known technically as a cold cathode indicator tube. It is composed of an anode and ten cathodes, made in the shape of the digits 0 to 9, and which are enclosed in a glass bulb or 'tube' containing neon gas. When the applied voltage is high enough (about 55 V) and one of the cathodes is earthed, the surrounding gas is ionized: this causes a glow around the outline of the cathode making the shape of the number visible. Whilst Nixie tubes have been superseded by LED displays, which use less electrical power, the 74141 IC serves as a good example of a decoder/driver with a higher than usual output level.

The 74141 has open collector outputs and handles up to 60 V at a maximum current of 7 mA. A logic 0 on one of the ten output lines lights up the corresponding decimal digit on the Nixie tube.

When higher currents are needed, the 7445 IC can be used. It can handle a maximum of 30 V at 80 mA, and has an open collector output. *Figure 54* shows an application where the 7445 drives a series of ten relays. The diodes are present to protect the IC from any back EMF the relays may generate when the supply current is turned off.

(See pages 294-295).

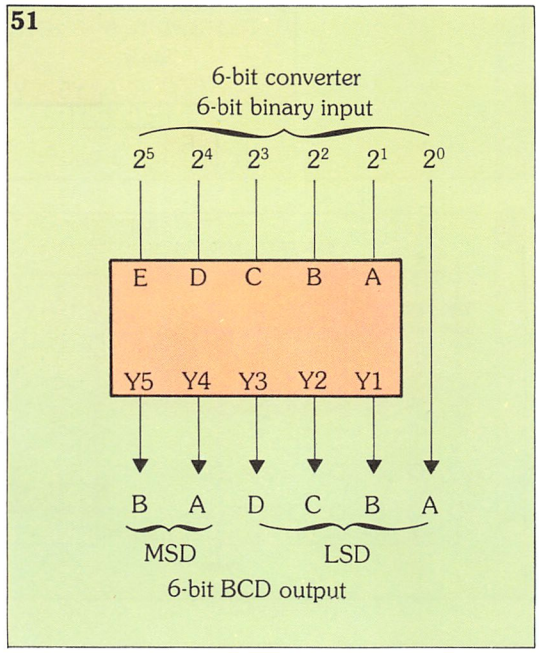
53. Nixie tube connected to a 74141 IC.

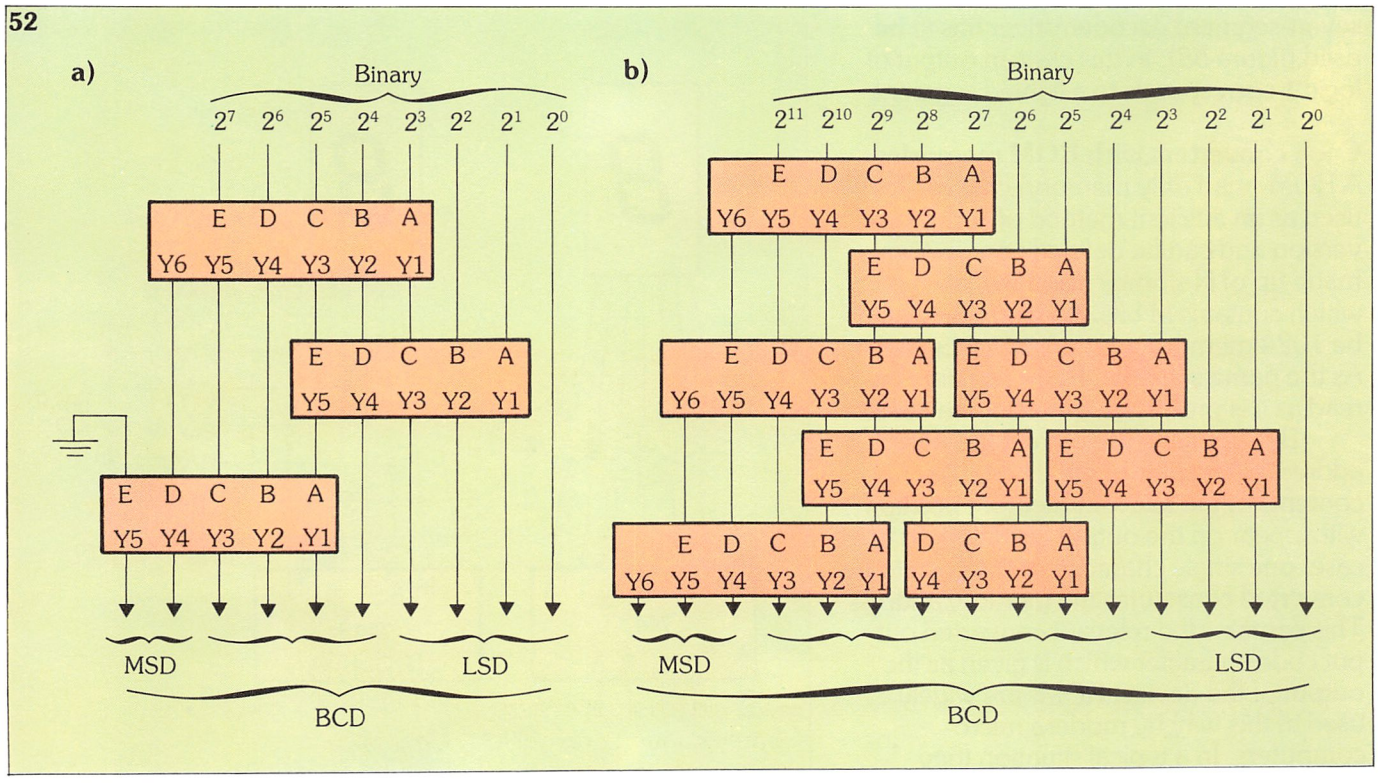
54. The 7445 IC as a relay driver.

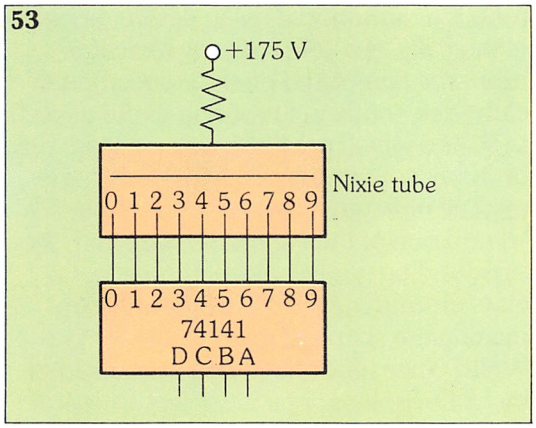
55, 56. Connection of a seven-segment display with either common anode, or common cathode respectively.

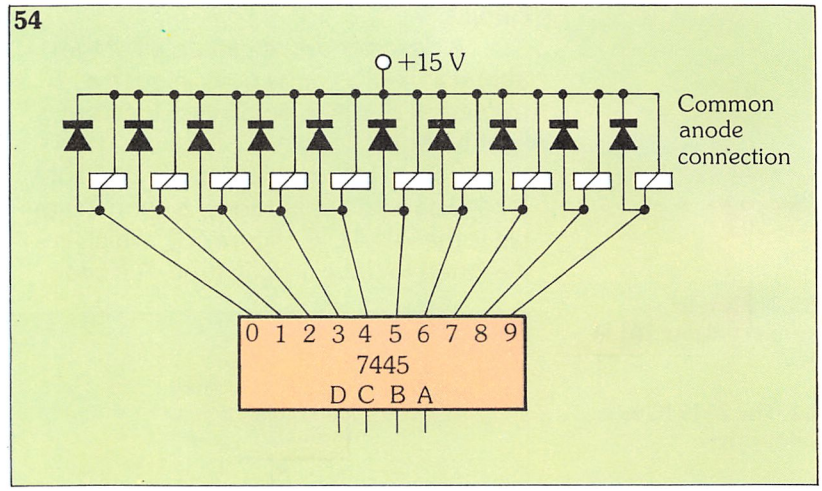
57, 58. Driving a common anode display with the 7447, and a common cathode display with the 7448 respectively.

59-62. Basic configuration of CMOS converters: 4555B; 4028; 4514B; 4511B, respectively.

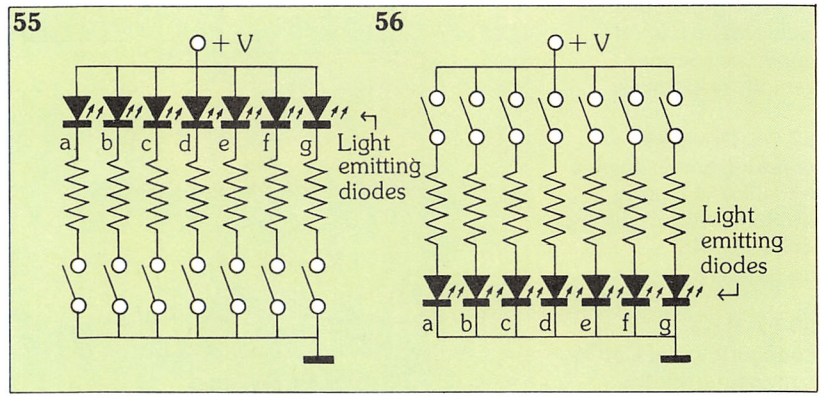


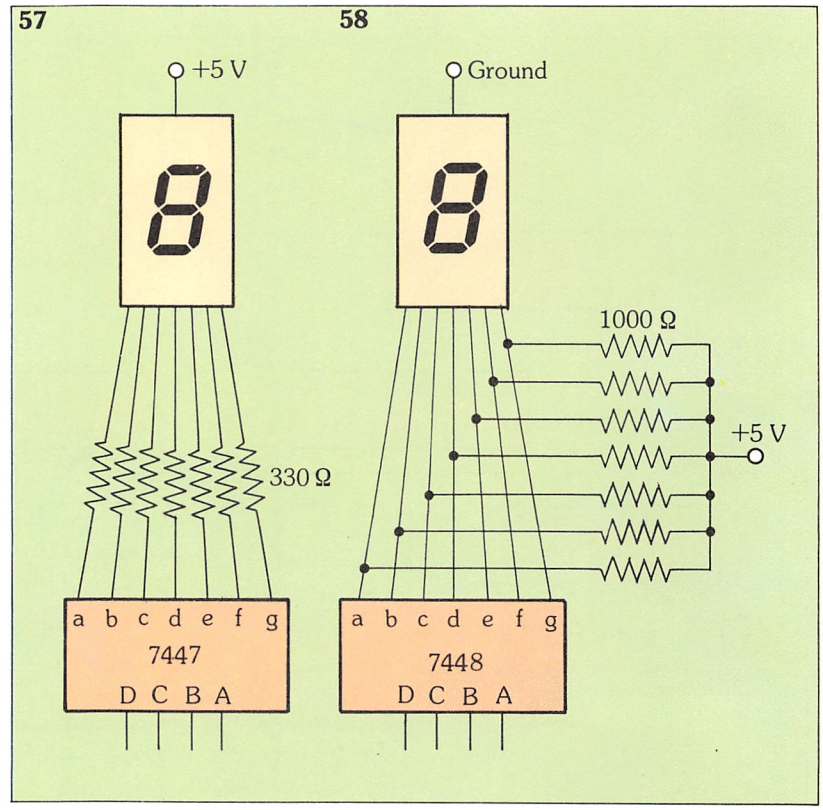






The LED is the most common light source used in seven-segment displays. Each LED is lit by a small direct current. The LEDs forming the display can be linked electrically in two ways: there can either be **common anode** (*figure 55*) or **common cathode** (*figure 56*) connections. The resistors in these circuits limit the current in each diode. When these LEDs are in common anode and connected to a decoder/driver (say the 7447) the open collector outputs take the place of the switches in the diagrams (*figure 57*) – logic 0 in an output illuminates the relevant segment. If the LEDs are in common cathode mode they cannot be directly driven by the 7447, and the 7448 BCD-to-seven-segment decoder/driver has to be used (*figure 58*). In this case an output of logic 1 corresponds to a lighted segment.

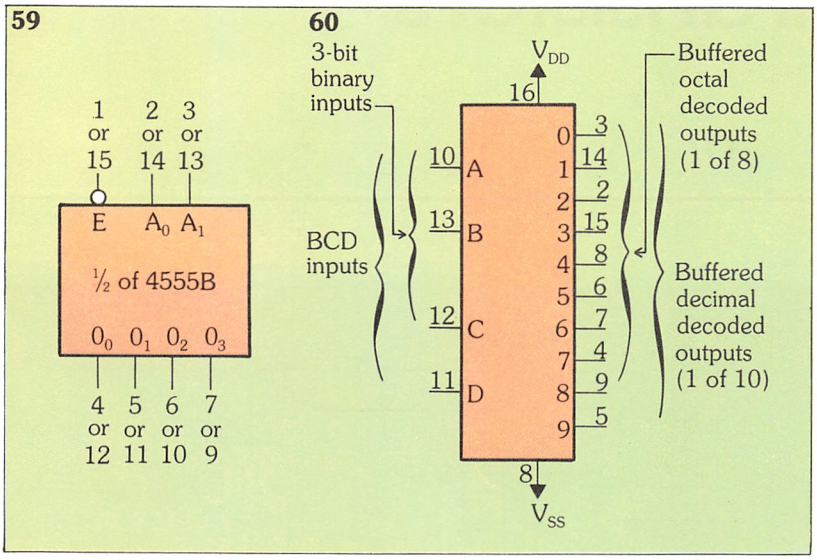


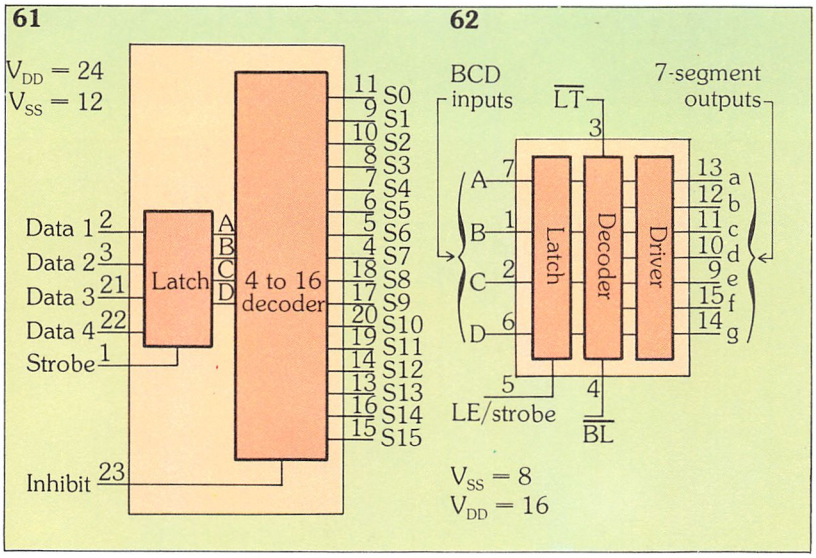


Code converters with ROM memories

A ROM (read only memory) chip can be used as an efficient method of code conversion and can be defined as a device made up of N storage positions, each of which contains M bits. An example would be 1024 memory positions of 8 bits each. As the name suggests, ROM can only be read as it is pre-programmed when made.

By supplying the correct memory address to the pins of the device, the contents of the relevant memory position will appear on the output terminals. In this case, one single character of the code to be converted constitutes the memory address. This contains the relevant converted output code character which is given as the output of the device. ROMs are widely used in this way by modern microcomputers. In a typical situation they





would convert the code generated by an alphanumeric keyboard into ASCII characters, and the ASCII characters into codes which enable visual display units and other alphanumeric output devices to work.

CMOS code converters

Because code converters are very important they are also available in CMOS ICs so that they can take advantage of CMOS' high operating speeds. Generally speaking, most of the previous points about the TTL devices apply to these chips.

Examples of CMOS code converters:

4555B – contains two independent 2 line to 4 line decoder/demultiplexers. Its terminal arrangement is shown in *figure 59*.

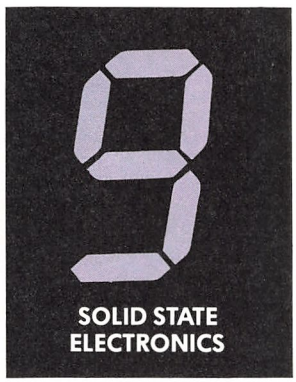
4028 – a BCD-to-decimal decoder. Also used as a 3 line to 8 line decoder/demultiplexer by using input D as an enabling input and disregarding outputs 8 and 9 (*figure 60*).

4514B – a 4 line to 16 line decoder/demultiplexer, shown in *figure 61*. The transition from 1 to 0 of the strobe input memorises the configuration of input signals in an internal latch. A logic 0 at pin 23 inhibits/disables the outputs.

4511B – a BCD-to-seven segment decoder/driver (*figure 62*) which is particularly suitable for driving LED displays arranged in common cathode mode. Each output can give a maximum current of 25 mA. Changing input LE from 0 to 1 memorises the combination of input signals in an internal latch. The six invalid inputs give a blank on the display.

Glossary

analysis	process of determining the function of a logic circuit
decoder/driver	piece of logic circuitry that can undertake code conversion operations and is capable of powering displays, relays etc.
latch	an electronic circuit that remains in a particular output state (1 or 0) until it receives an input signal. The output condition may then change to the opposite state
ROM code converter	code conversion device that uses the input code to address the storage position that holds the relevant output code, which is given as output
synthesis	process of designing a logic circuit to carry out a specific operation or operations



Transistors

What a transistor does

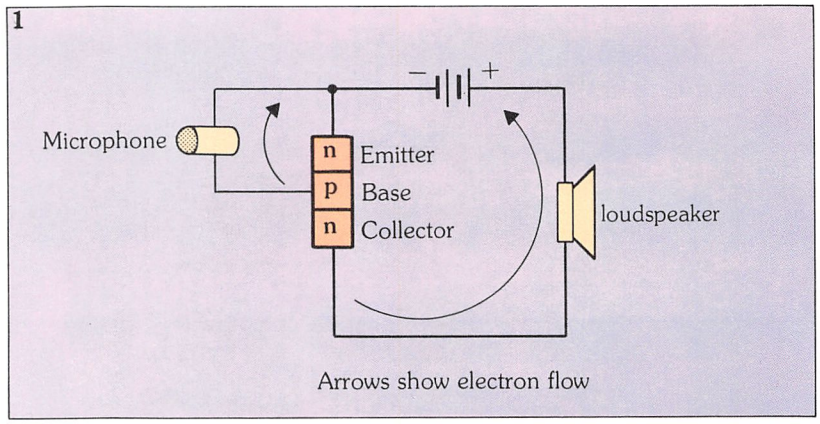
A transistor can be used either as an amplifier or as a switch. This section looks in detail at the make-up of transistors and how they work, but let's start by recapping on these two basic functions. Both of the diagrams, *figures 1* and *2*, refer back to similar examples explained in *Solid State Electronics 2*.

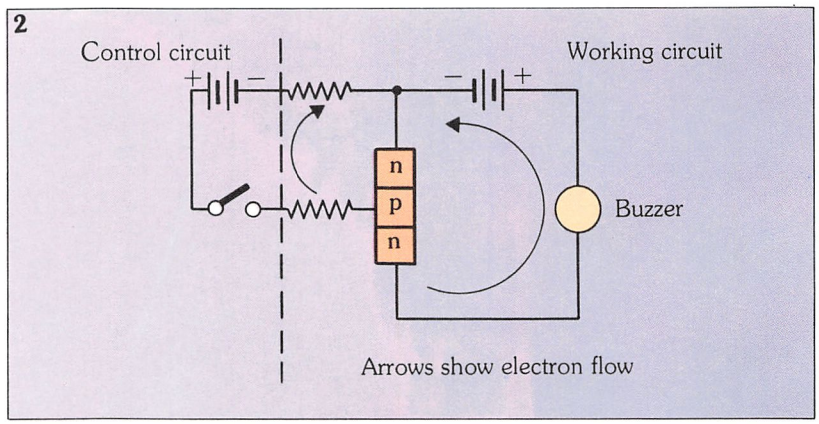
Figure 1 shows how a transistor can be used as an amplifier. Here, an n-p-n transistor amplifies the signal (current) from the microphone to a high enough level to drive the loudspeaker.

The microphone converts the power of the sound waves into an electrical waveform. When there is no sound, the transistor blocks the current from the power supply to the loudspeaker; when there is sound, the microphone's electrical output draws electrons from the base (p-region) of the transistor. The greater the sound, the more electrons that will flow from the emitter to the collector, and through the circuit to the loudspeaker.

The transistor is needed because the microphone's output is too low to drive the loudspeaker. More power is therefore needed which, in this case, is supplied by the battery. The transistor regulates the flow of electrons from the battery producing an amplified (enlarged) copy of the microphone's signal.

Looking now at the switching function, *figure 2* shows the telegraph sending and receiving circuit. The transistor is necessary because the signal from the switch is too weak (due to the resistance of the long wires) to operate the buzzer. When the control voltage exceeds the **switching threshold**, electrons are drawn out of the p-region of the transistor. The transistor then starts conducting a large quantity of electrons from the power supply, thereby activating the buzzer-battery circuit.





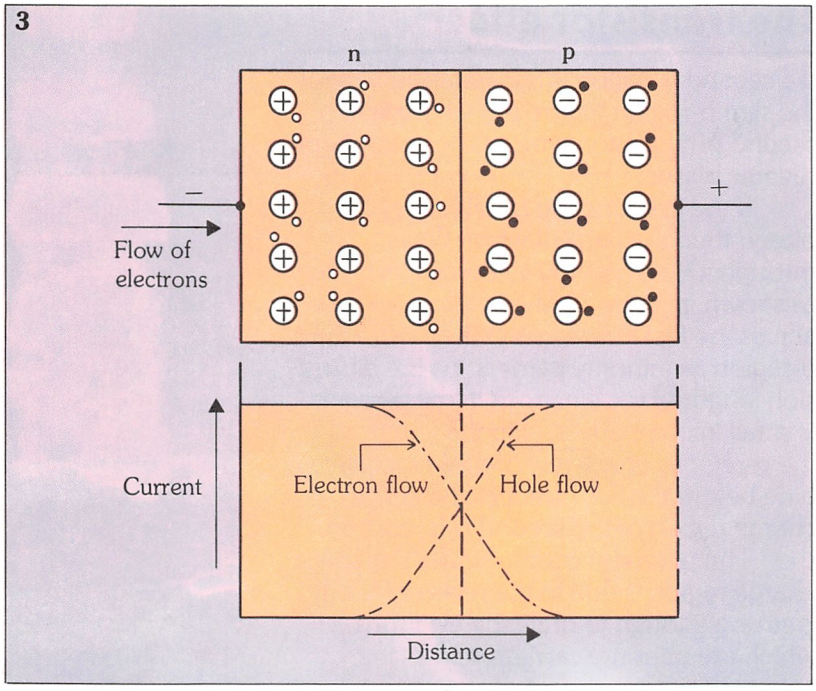
1. Using an n-p-n transistor as an amplifier.

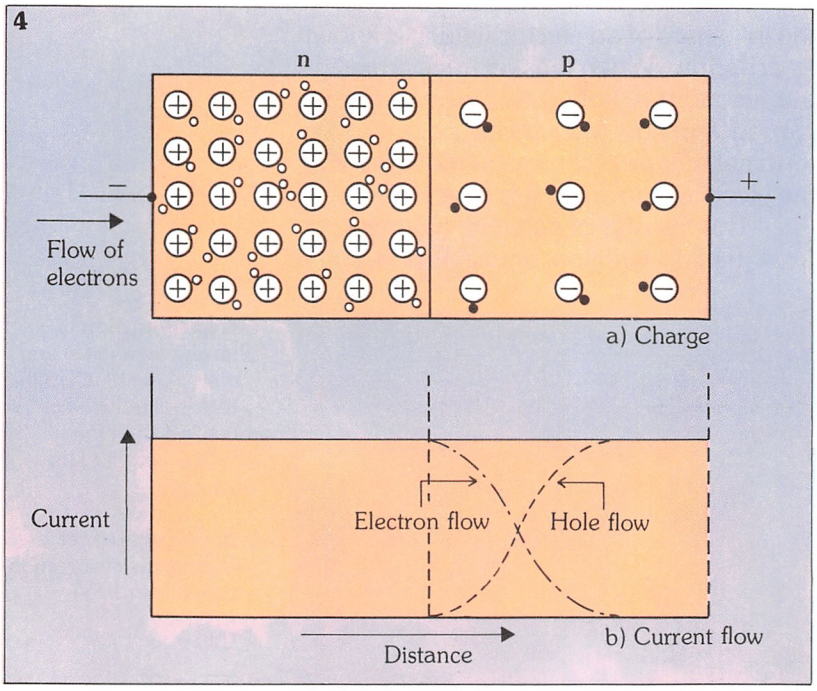
2. The switching function of an n-p-n transistor.

The threshold voltage of a silicon transistor is about 0.6 V – the same as that of a silicon diode. When the supply voltage exceeds 0.6 V, electrons are withdrawn from the p-region and the transistor switches 'on'. It will remain 'on' as long as sufficient electrons are being withdrawn from the base. The current in the circuit that is being operated by the transistor can be varied across a range proportionally to the applied **control** current.

The difference between amplifying and switching transistors

Transistors are generally classified either as switches or amplifers, but not both. This is because, although they can perform both





3. Current flow in a directly polarised, symmetrical p-n junction.

4. Concentration of charge carriers (a) and current flow **(b)** in a forward biased asymmetric p-n junction.

functions, they do so at the expense of performance. It is the circuit controlling the transistor that determines its function.

Asymmetric p-n junctions

A useful starting point for analysing and understanding the operation of a transistor is to recall how current flows in a directly polarised p-n junction. *Figure 3* illustrates the type of current flow in a symmetrical junction where the concentration of carriers is equal on both sides of the junction. Now consider what happens when there is an asymmetric concentration of charge carriers. Suppose that there is a high concentration of electrons in the n-region and a low concentration of holes in the p-region. This is shown in *figure 4a*.

If a forward bias is applied, the electrons coming from the n-region cross the junction to the p-region where they will meet only a few holes: they will therefore travel a considerable distance before they are all recombined. At the same time, those holes flowing across the junction in the opposite direction (from the p to the n-region) will meet a high concentration of electrons, and will recombine quickly without moving away from the junction. The distribution of electron and hole currents is shown in *figure 4b*.

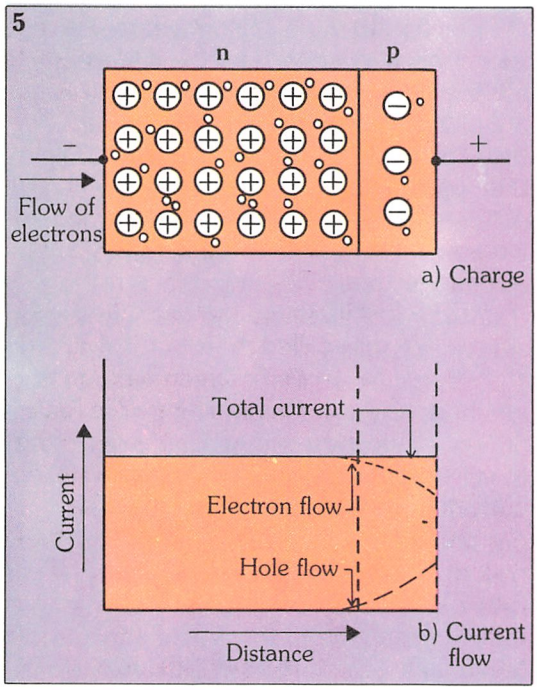
You can see that in the vicinity of the junction, all the current is composed of the flow of electrons from the highly concentrated n-region, with only a small hole current from the low concentration p-region. This is described as a **high injection of electrons** into the p-region. The ratio between the electron current and the entire current which flows across the junction is called the **injection efficiency**. In asymmetrical junctions this can reach 99%.

As we have said, in the p-region near the junction most of the current is due to a flow of electrons. Once these electrons have crossed the junction, they become minority carriers in this region.

The average distance covered by minority carriers before they recombine is called the **diffusion length**. The diffusion length depends upon the mobility and lifetime of the minority carriers. This is generally inversely proportional to the concentration of the minority carriers in the semiconductor material. The diffusion length of electrons in the low concentration p-region is greater than that of holes in the high concentration n-region. The significance of diffusion length in transistors will be seen later on.

If the length of the p-region is reduced to much less than the diffusion length of the electrons (*figure 5a*), only a few of the electrons injected into the p-region will recombine before reaching

the terminal wire; the remainder will flow on through the external circuit. The flow of electrons and holes through the structure is shown in *figure 5b*. The important point to remember in this case is that the current which crosses this slim p-region is almost completely composed of a flow of electrons. These electrons are minority carriers in this region.



The transistor effect

If a second n-region is added to the right of the slim p-region (*figure 6a*) we have a second p-n junction which is normally kept reverse biased.

If the first junction is still forward biased then the current which flows through it is still mainly a flow of electrons, as shown in *figure 5*. These electrons flow across the first junction and then cross the p-region as minority carriers. As the diffusion length of the electrons in the p-region is small in comparison with the length of the p-region, only a few of these recombine before the flow reaches the space charge region of the second p-n junction.

The potential energy across this space charge region is due to the reverse bias. It is strong enough to draw the electrons, which are minority carriers, across the junction to the second n-region. Since the flow across the second junction is under the influence of an electric field, it is known as **drift flow**. When the electrons arrive in the second n-region, they become majority carriers and flow towards the positive terminal. *Figure 6* shows the movement of the electron current in this structure.

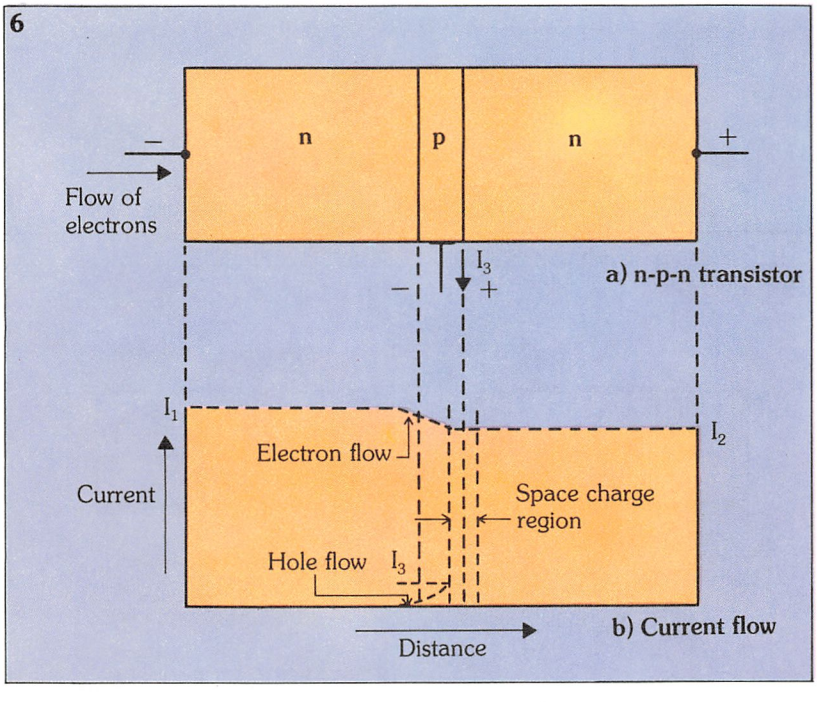
This transfer of negative electrons from the first n-region, across a narrow

5. Concentration of charge carriers (a) and current flow **(b)** when the p-region is reduced to much less then the diffusion length of the electrons.

Left: one of the first pocket transistor radios – available in 1954.

p-region and a reverse biased junction into a second n-region, makes up a **transistor effect**. As some electrons recombine in the p-region, the electron current which flows across the second n-region is slightly less than the current that flows through the first n-region. By applying a low voltage to the first forward biased p-n junction, a current will flow in this low resistance circuit. This current then passes through the second p-n junction (a high resistance circuit) allowing a higher output voltage to be obtained. Since the applied voltage can be varied over a range of values, the output voltage will vary proportionally with it. In this way, either a switching or an amplifying function is obtained.

6. The movement of the electron current comprising the transistor effect.



p-n-p junction transistors

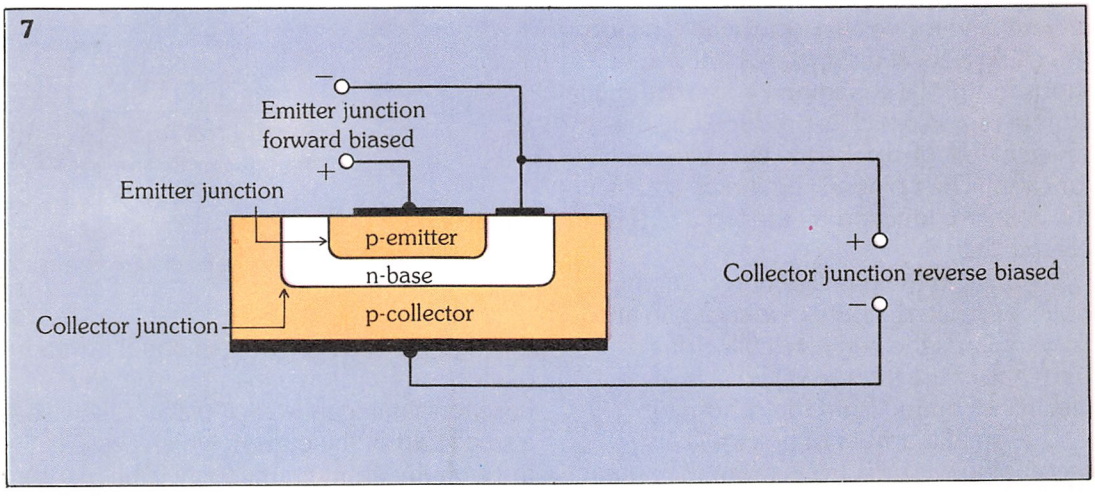
As you will have gathered, a transistor consists of two p-n junctions in one of two configurations: either p-n-p or n-p-n. The arrangement of a typical p-n-p transistor is shown in *figure 7*. This arrangement is called a **planar structure**, because the different semiconductor materials (n and p-type) are in layers or **planes**. In p-n-p transistors, the first, positively biased p-region is called the emitter because it emits (injects) holes along the forward biased junction into the central n-region. The narrow n-region is called the **base**. The other p-region is known as the **collector**. It is negatively biased to collect the holes coming from the central region, across the second reverse biased junction.

The first junction, between the emitter and base, is called the emitter junction; the second junction, between base and collector, the collector junction. We have seen that the concentration of holes in the p-emitter must be greater than the concentration of electrons in the n-base. The emitter is doped much *more* than the base region. Later on we will see why the collector region is doped *less* than the base.

Current flow in a p-n-p transistor

For the moment we will look at a biased collector junction. First consider what happens when a negative voltage is applied to the collector (while no voltage is applied to the emitter). This causes the **collector junction** to be reverse biased (*figure 8a*). No current will flow through the emitter junction. Current can only flow through

7. The planar structure of a typical p-n-p transistor.



the collector junction and consists of minority carriers produced by the thermal agitation in and around its space charge region. This is the **reverse saturation current** of the collector junction.

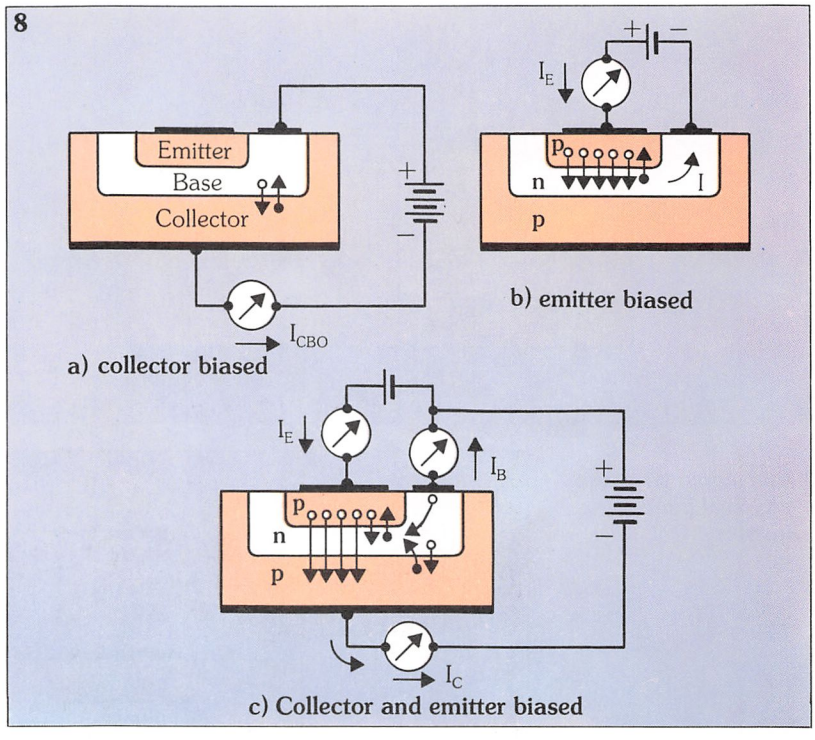
In junction transistors this reverse saturation current is known as I_{CBO}. The subscript letters CBO indicate that the current flows from the **collector** towards the **base** and that the emitter is **open.** Since I_{CBO} is due to thermal agitation, it will increase with any rise in temperature and is therefore said to be **temperature sensitive.** At any given temperature, I_{CBO} will remain constant, independent of the collector voltage, as long as the voltage is greater than about 1 V and stays below the avalanche breakdown level of the junction.

Secondly, look at what happens when a forward voltage is applied to the emitter junction and no voltage is applied to the collector *(figure 8b)*. As we have previously seen, a small forward voltage (usually less than 1 V) is enough to cause a strong current flow through a junction. Once the holes enter the base region they become minority carriers. As there is no connection with the collector region, the holes cannot flow through the collector junction, so they flow from the base region towards the negative electrode connected to the base.

Finally let's consider what happens when the emitter junction is foward biased while the collector junction is reverse biased. This is the transistor's normal operating condition and is shown in *figure 8c*. The hole current will flow across the emitter junction to the base region as before. The holes will become minority carriers and move away from the junction, travelling a short distance – a fraction of thousandth of a centimetre – into the space charge region which is produced by the reverse voltage applied to the collector junction. The holes will be drawn across the collector junction by the force of this electric field.

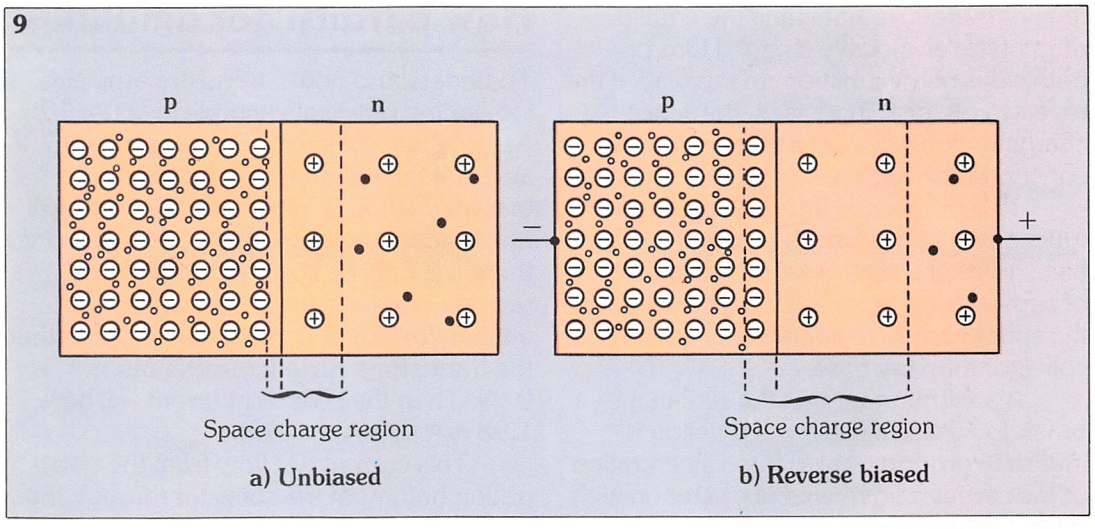
Once in the p-collector region, the holes become majority carriers again and flow towards the negative collector terminal where they are filled with electrons coming from the conductor.

A small number of holes will also recombine with electrons when they pass through the base region. An equal number of electrons will have to enter from the base terminal to take their place. Likewise a small part of the current which passes through the emitter junction will be made



8. Current flow in a p-n-p transistor:
(a) collector biased;
(b) emitter biased;
(c) collector and emitter biased.

Left: examples of transistors. Note how small they are compared with the pen. (Photo: Motorola).



9. Charge concentration in an asymmetric p-n junction which is:
(a) unbiased;
(b) reverse biased.

up of electrons which flow from the base to the emitter. These electrons must also move to the base region from the base terminal.

The sum of these two small streams of electrons constitutes the **base current**, I_B. The current that flows in the emitter terminal is known as the **emitter current**, I_E, while that in the collector wire is called the **collector current**, I_C.

Remember, the flow of holes only exists within the semiconductor material. The flow of holes in one direction is simply another way of saying that the bound electrons move in the other direction.

Since the total number of electrons which flow outside the transistor's structure must be equal to those which enter it, the sum of the collector current, I_C, and the base current, I_B, must be equal to the emitter current, I_E. Therefore we can say:

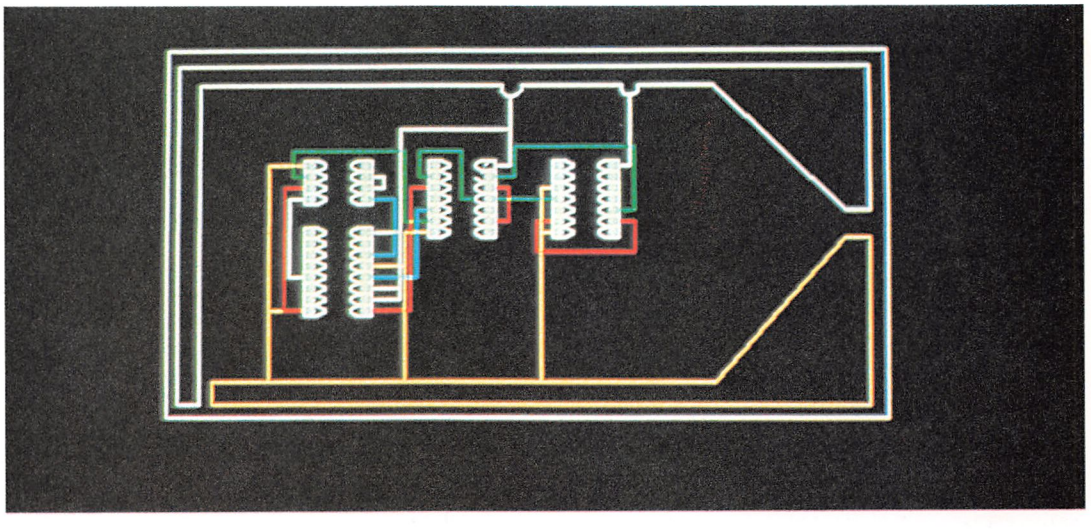
$$I_E = I_C + I_B$$

Collector junctions

In a symmetrical p-n junction the space charge region is also symmetrical on both sides of the junction. Now consider the non-polarised asymmetrical junction as shown in *Figure 9a*. In establishing a condition of equilibrium, the initial spread of holes across the p-region will leave just a thin layer of negative ions behind. The corresponding flow of electrons from the n-side will leave a relatively thick layer of positive ions. The barrier will therefore be asymmetrical and the space charge region will be wider on the low concentration side of the junction.

If a reverse voltage is applied to the junction (*figure 9b*), electrons in the n-region move away from the junction to a greater extent than the holes in the p-

Right: a circuit diagram shown on a colour graphics terminal. (Photo: Tektronix)



region, thereby maintaining the equilibrium of the electrically charged ions on both sides of the junction. In this way, if the reverse voltage is increased, the space charge region will exist mainly in the low concentration region.

This property is important to the functioning of the junction transistor. The base normally has a greater concentration of carriers than the collector. This makes the space charge region spread more in the collector than the base.

Also remember that the avalanche breakdown voltage in a p-n junction is inversely proportional to the concentration of the carriers. So to enable operation with a high collector voltage, a low enough level of carriers in the collector region must be established. A higher concentration will therefore be used in the base region, and will move in the collector region of the junction.

In addition, we have seen that for a high emitter injection efficiency, the concentration of majority carriers in the emitter must be large compared with that in the base; we now see that it must also be large in comparison with the concentration in the collector.

Current transfer ratio

As shown earlier, the collector current is almost equal to the emitter current. If the emitter current is varied slightly, the collector current will change along with it. The ratio between the variation of the collector and emitter currents is called the **current transfer ratio** and is indicated by the Greek letter $\propto$. We can therefore say that:

$$\propto = \frac{\Delta I_C}{\Delta I_E}$$

where Δ (delta) stands for the variation (or change) in the current. The value of $\propto$ depends on two factors: the injection efficiency of the emitter junction, and the transport efficiency of the base region. This latter property is the ratio between the number of minority carriers which reach the collector and the number of minority carriers injected into the base region through the emitter junction. In practice, $\propto$ can be of any value but is usually very slightly less than 1, typical values being 0.98 – 0.99.

How a transistor amplifies

To understand how a transistor amplifies, look at the practical example in *figure 10*. Assume that the emitter voltage V_E is 0.7 V and that the resistance of the forward biased junction is 50 Ω. As the emitter will not conduct until its voltage exceeds 0.6 V, there will only be 0.1 V available to provide emitter current; this will give an emitter current of 2 mA. Then suppose that the transistor's current transfer ratio, $\propto$, is 0.99. Then the collector current will be 1.98 mA (2 mA × 0.99).

This current will flow from the polarisation battery of the collector through the load resistance of 5000 Ω, causing a drop of 9.9 V across the resistor. This voltage drop will give a collector voltage of 5.1 V (15 V – 9.9 V) across the collector junction of the transistor.

Now suppose the emitter voltage is modified by 1 mV. The change in the emitter current will be 20 μA (1 mV ÷ 50 Ω) and that of the collector current will be 19.8 μA (20 μA × 0.99). This change in current will cause a variation in voltage of 99 mV across the 5000 Ω load. We can therefore see that a variation of 1 mV in the emitter voltage will produce a variation of 99 mV across the circuit resistance, in other words an amplification of 99.

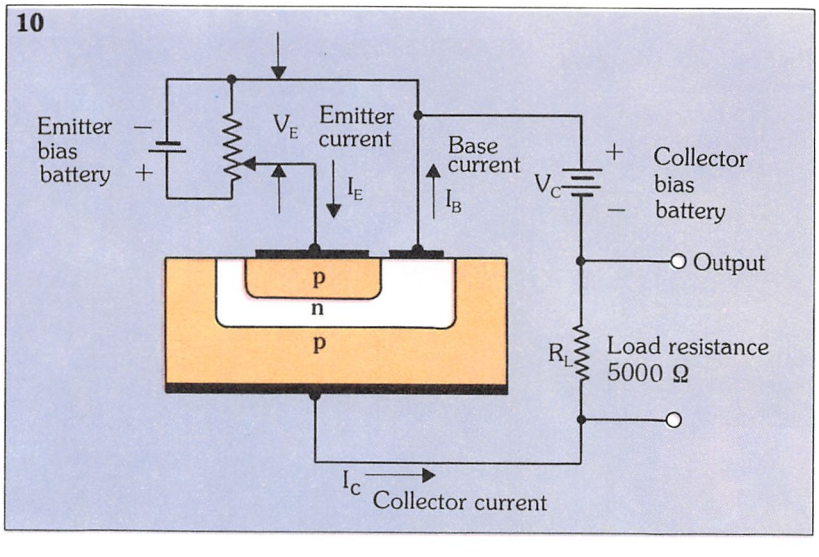
Later on you will see how modifications to the transistor's connections to the external circuit will lead to amplification of voltage, amplification of the current, or both.

10. A practical example of how a transistor functions as an amplifier, producing an amplification of voltage from 1 mV to 99 mV.

11. Configuration of an n-p-n transistor illustrating the arrangement of the different polarities.

12. An amplifying circuit using both p-n-p and n-p-n transistors.

Below right: the first silicon transistor produced on an industrial scale by Texas Instruments in 1954.



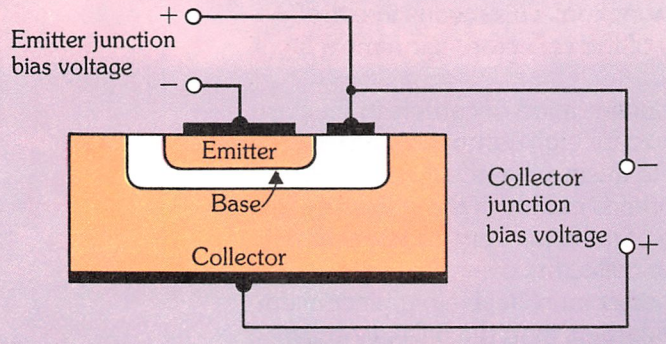
n-p-n junction transistors

n-p-n and p-n-p transistors work in exactly the same way: they both switch and amplify electricity. However, they are exactly *opposite* each other in construction and operation. Like p-n-p transistors, n-p-n transistors have two p-n junctions, but these are made from two layers of n-type material separated by a thin p-region. In n-p-n transistors the emitter junction is forward biased and the collector junction is reverse biased as in p-n-p transistors.

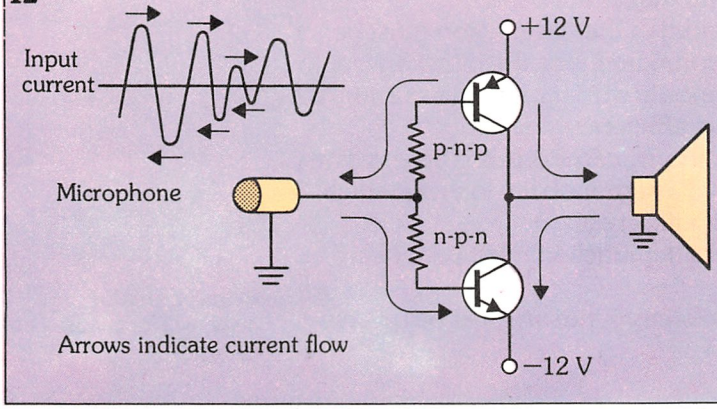
There are two substantial differences to remember: firstly, the operating voltages will be opposite to those of a p-n-p transistor. The n-type emitter must be negative with respect to the p-type base so as to be forward biased. The n-type collector must be positive with respect to the base, so as to be reverse biased. Secondly, the n-emitter will now inject electrons into the p-base which will become minority carriers spreading through the base region until they are drawn across the collector junction by the electrical field produced by the positive potential. *Figure 11* shows how the polarities are arranged.

n-p-n transistors are both faster operating and cheaper to manufacture than p-n-p types, and so are more widely used.

11

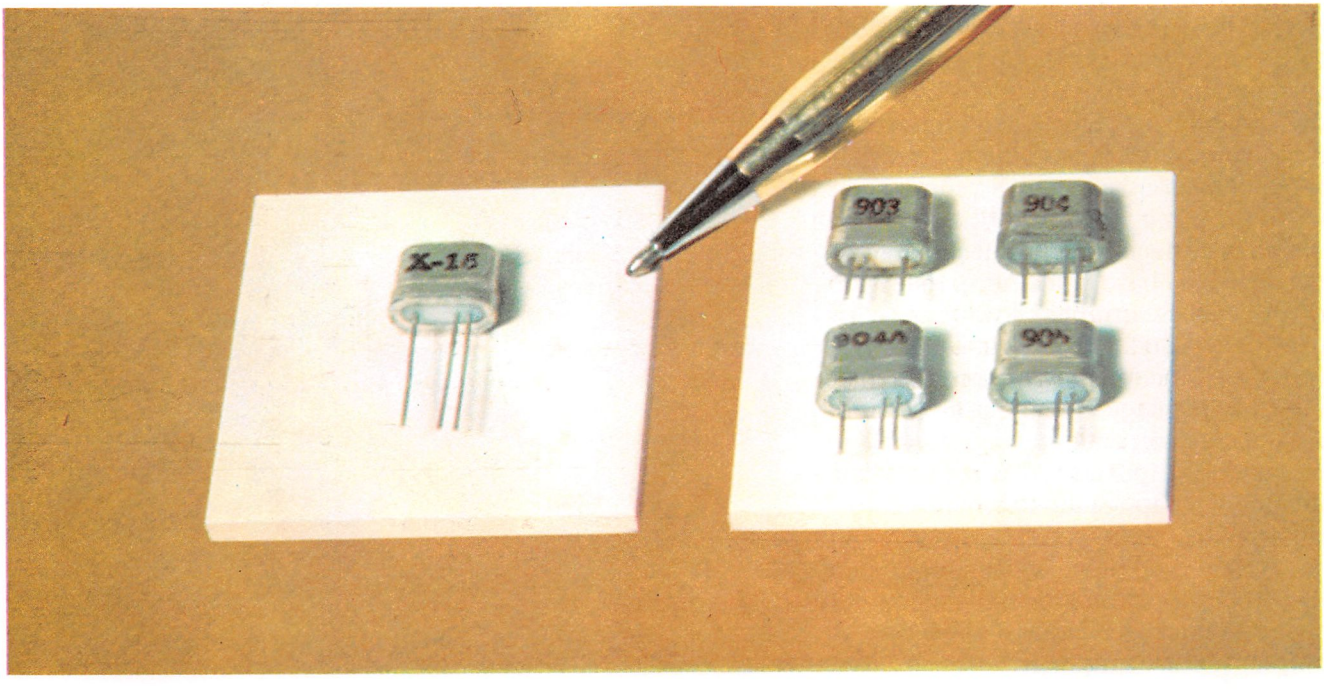


12



Using p-n-p and n-p-n transistors together

The opposite characteristics of these two types of transistor can be combined and usefully exploited. An amplifying circuit using both an n-p-n and a p-n-p transistor



is shown in *figure 12*. To find out why this is an improvement, think back to the original circuit that used a single n-p-n transistor (*figure 1*).

Sound waves striking the microphone generate current waves in the wire leading to the transistor base. (Remember current always flows *into* the base of an n-p-n transistor.) The transistor then amplifies these waves by regulating the current supply and this amplified current drives the loudspeaker which reproduces the sound.

Now let's look at what's wrong with this circuit. Sound waves are in effect alternating air currents, so any audio amplification system must be capable of working with alternating electric current. A typical microphone draws current in and out of the amplifying circuit, which in turn has to move the loudspeaker cone in (push) and out (pull) to mimic the movements of the microphone diaphragm thus reproducing the original sound wave.

The one transistor circuit has only been doing half this job. As it has only been able to have current pumped *into* the base of the transistor, it has only been able to produce the positive half of the sound waves. Resistors and capacitors can modify this circuit to give an a.c. output (but it is very inefficient) and it is known as a **class A amplifier**.

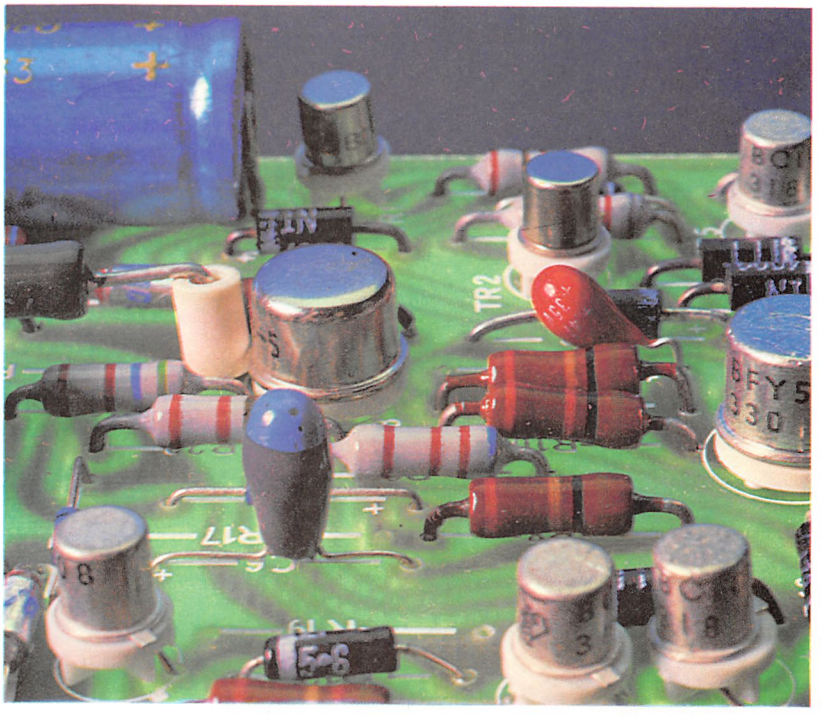
Figure 12 shows a **class B amplifier**, also known as a **push-pull amplifier**. This refers to the fact that it uses two transistors to provide an alternating current. When the input voltage goes positive, current can flow into the base of the n-p-n transistor and so current will flow through its collector out of the loudspeaker. No current can flow in the p-n-p transistor. When the input voltage goes negative, current can only flow out of the base of the p-n-p transistor and this causes current to flow through its collector into the loudspeaker. Thus, the n-p-n transistor amplifies the positive part of the input voltage, and the p-n-p transistor similarly deals with the negative half. In this way the output current is an amplified and inverted copy of the alternating current from the microphone.

A class B amplifier which uses an n-p-n and p-n-p transistor in the manner shown in *figure 12* is called a **class B complementary amplifier.**

Structural requirements of a transistor

1. The concentration of carriers in the collector region must be low, so as to obtain a high breakdown voltage at the collector junction. This means that the resistivity of the collector region must be high.
2. The concentration of carriers in the base region must be high compared with the collector to ensure that the space charge region spreads mainly in the collector. The resistivity of the base must be lower than that of the collector.
3. The emitter must have a higher concentration of carriers than the base, to ensure a **high injection efficiency**. This means that the resistivity of the emitter must be much lower than that of the base.
4. The width of the base region must be narrow compared with the diffusion length of the minority carriers. This gives a **high transport efficiency**.
5. The minority carriers in the base region must have a high mobility to ensure high transport efficiency.
6. The emitter junction must be forward biased.
7. The collector junction must be reverse biased.

Below: a smoke detector transistor circuit.



Science Photo Library/H. Schneebeli

Circuit Symbols

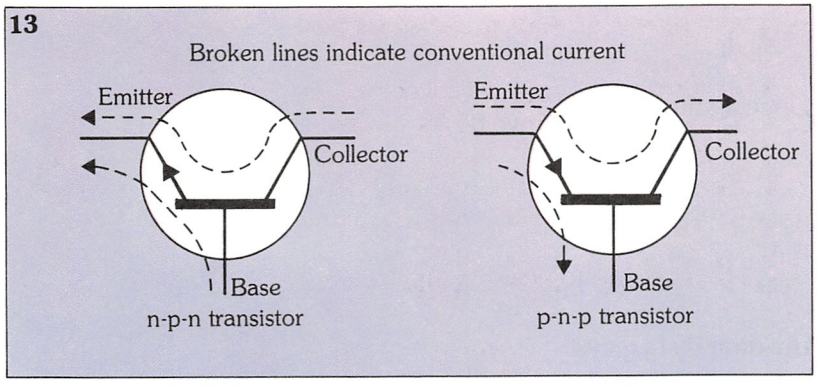
The conventional symbols for n-p-n and p-n-p transistors are shown in *figure 13*.

The difference between p-n-p and n-p-n transistors becomes even clearer if the logic behind the two symbols is examined. In this case, it is better to think in terms of conventional current rather than motion of electrons. Looking at the n-p-n symbol, you can see that the transistor is controlled by a small current pumped into the base when electrons are withdrawn. The result of this is a large current flow from collector to emitter. The arrow in the symbol serves as a reminder of both the control current and the working current. It shows the movement of current away from the base and collector and out of the emitter.

Remember, an arrowhead in the symbol for any semiconductor device indicates a p-n junction pointing from to p to n. Conventional current passes from p to n but is blocked in the other direction.

In the p-n-p transistor symbol the arrowhead on the emitter is pointing towards the base. This is because the base is n-type and the arrow indicates the current flow from p to n. The control current flows from emitter to base; the working current from emitter to collector.

13. Conventional circuit symbols for the n-p-n and p-n-p transistors.



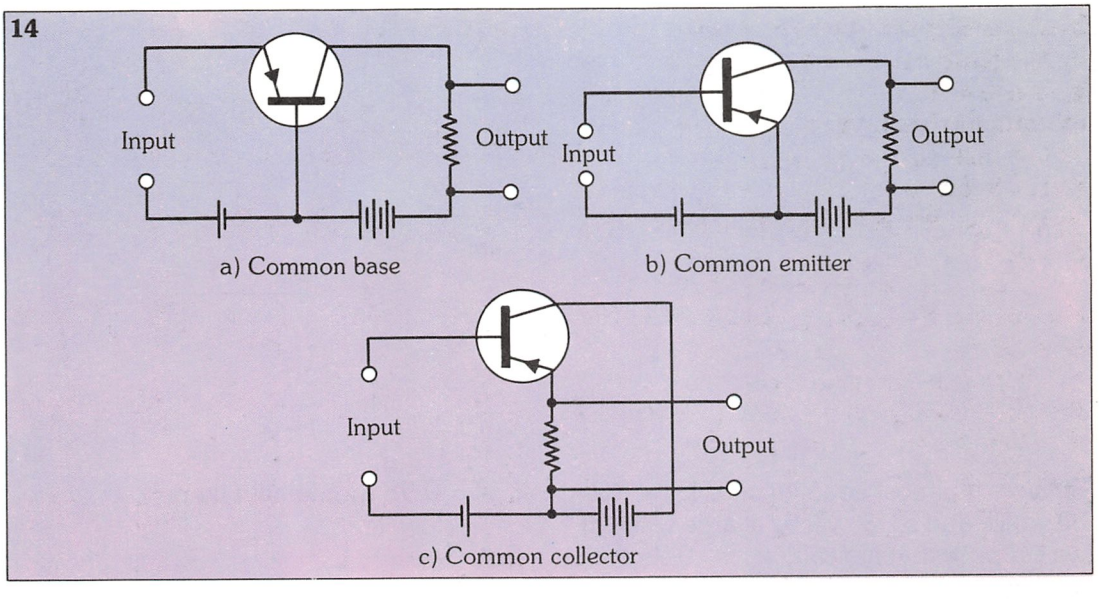
Circuit arrangements

It is possible to connect a transistor so that either its emitter, collector or base is shared by the input and output circuits. These are known as **common** connections.

Common base circuits

The common base circuit in *figure 14a* has the same configuration as shown in *figure 10*, but it has been redrawn using circuit symbols. Remember, in a common base circuit the current transfer ratio between input and output ($\propto$) is slightly less than one, and a high voltage amplification can be obtained.

14. Configurations for transistor circuits:
(a) common base;
(b) common emitter;
(c) common collector.



Common emitter circuits

This arrangement is shown in *figure 14b*. The input is connected to the base and the output is taken from the collector, while the emitter is common to both circuits. It will be useful to work out the new base to collector current transfer ratio. This is usually represented by the symbol β, and:

$$\beta = \frac{\Delta I_C}{\Delta I_B}$$

Considering the relationship between the currents in a transistor, β can be calculated by starting from α, the common base current transfer ratio. We have seen that:

$$\Delta I_E = \Delta I_B + \Delta I_C$$

and also that:

$$\alpha = \frac{\Delta I_C}{\Delta I_E}$$

by substitution we obtain:

$$\frac{\Delta I_C}{\alpha} = \frac{\Delta I_C}{\beta} + \Delta I_C$$

or dividing throughout by ΔI_C:

$$\frac{1}{\alpha} = \frac{1}{\beta} + 1$$

rearranging we have:

$$\frac{1}{\beta} = \frac{1}{\alpha} - 1$$

$$= \frac{1 - \alpha}{\alpha}$$

$$\beta = \frac{\alpha}{1-\alpha}$$

Since α is almost equal to one, $1-\alpha$ will be small, and so β will be a large value. If we suppose that typically $\alpha = 0.98$ then:

$$\beta = \frac{0.98}{1-0.98} = 49$$

Therefore the common emitter circuit gives another current transfer ratio between the input (base) and the output (collector). Because of this, β is normally called the **current gain**.

The common emitter circuit arrangement is the most useful and widely used. Besides giving a high current gain it also supplies a voltage amplification which will be discussed in a later chapter.

Common collector circuits

This method of connection is shown in *figure 14c*. The input goes to the base and the output comes from the emitter.

The current transfer ratio from the base to the emitter needs to be examined in this case. The transfer ratio is:

$$\frac{\Delta I_C}{\Delta I_E}$$

Once again:

$$\alpha = \frac{\Delta I_C}{\Delta I_E}$$

$$\Delta I_E = \Delta I_B + \Delta I_C$$

Dividing by I_B gives:

$$\frac{\Delta I_E}{\Delta I_B} = 1 + \frac{\Delta I_C}{\Delta I_B}$$

$$= 1 + \beta$$

$$= 1 + \frac{\alpha}{1-\alpha}$$

$$= \frac{1 - \alpha + \alpha}{1 - \alpha}$$

$$= \frac{1}{1-\alpha}$$

If $\alpha = 0.98$ (nominally) then:

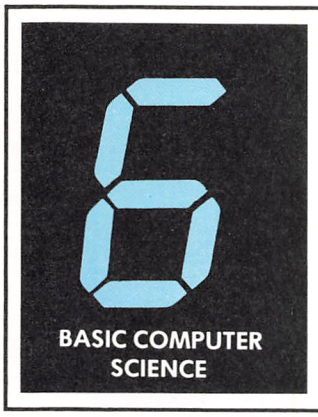
$$\frac{\Delta I_E}{\Delta I_B} = 50$$

The current gain of this connection is almost the same as the common emitter current gain. In this common collector circuit however, the output voltage will always be less than the input voltage, due to the very small drop in voltage between the base and the emitter of the transistor (see *figure 13c*).

These different transistor configurations will be dealt with in greater detail in the next few chapters.

Glossary

base	the region in a transistor between the emitter and the collector, into which minority carriers are injected
bipolar transistor	a 'normal transistor', that is to say one that utilises charge carriers of both polarities (as opposed to a field effect transistor)
class A amplifier	a single transistor amplifier used in an inefficient way to amplify AC signals
class B amplifier	a combination of two transistors in a complementary or push-pull configuration
collector current (I_C)	the current which flows into the collector of an n-p-n transistor, or out of the collector of a p-n-p transistor when in circuit
collector	the region in a transistor into which carriers flow from the base through the collector junction
collector junction	normally reverse biased junction between the emitter and base of a transistor. The current which flows through this junction is controlled by the introduction of minority carriers into the base
current transfer ratio (common base)	indicated by the sign $\propto$. Shows the ratio of the variation in emitter and collector currents: $\propto = \frac{\Delta I_C}{\Delta I_E}$
diffusion length	average distance that a minority carrier can travel before it recombines with an electron
drift flow	the flow of charge carriers across the second junction of a transistor
emitter	the region in a transistor from which carriers flow, through the emitter junction into the base
emitter current (I_E)	the current which flows out of the emitter of an n-p-n transistor or into the emitter of a p-n-p transistor
emitter junction	normally forward biased junction between the emitter and base of a transistor. Minority carriers flow through this junction from the emitter to the base
I_{CBO}	reverse saturation current of the collector junction. The current flows from the Collector towards the Base with the emitter is Open
planar structure	describes the way in which transistors are constructed – with planes of semiconductor material fused one onto the other
switching threshold	the voltage at which a forward biased p-n junction begins to conduct (about 0.6 V for a silicon junction)



Storing data structures

Data structures

A computer's function is to process data in the way that it is instructed by a program. Data does not exist as an amorphous mass but must be arranged into structured sets that the computer can handle.

When structuring a set of data we need to understand the connections that exist between the data items, and their relation to the problem to be solved. The operations to be performed on the data items, and the frequency at which these operations occur, will help to decide which *type* of data structure to use and how to write the computer program to handle it.

This chapter looks at some of the ways that data structures can be used by computers, and how these data sets are held in memory.

The most basic unit of data is known as an **element**, or an elementary **field**. The **field length** determines how many characters the data element is made up of. At its simplest, an element of data can be only one character long. In this case we would say that the field length was equal to 1.

The simplest type of data is the single element. An example of this is the data which might serve as the input to a program which calculates the square root of numbers. The computer might show on its display 'Type in a number'. If we then type in a number, say 16, we have given the program its single data element. The computer might then give us the answer in the form 'The square root of 16 is 4'.

If the program is more complex, for example, one that will plot a temperature graph and show the maximum and mini-

Left: it is possible to graphically reconstruct data held in computer memory on a VDU. In this example, the contour lines of a map, and information regarding rivers and the road and rail network are shown.

mum temperatures over a period of time, a **set** of data elements would be needed. In this case, the set could be temperature readings which were taken every thirty minutes. This data, which constitutes a finished set, must be stored to be used in the execution of the program.

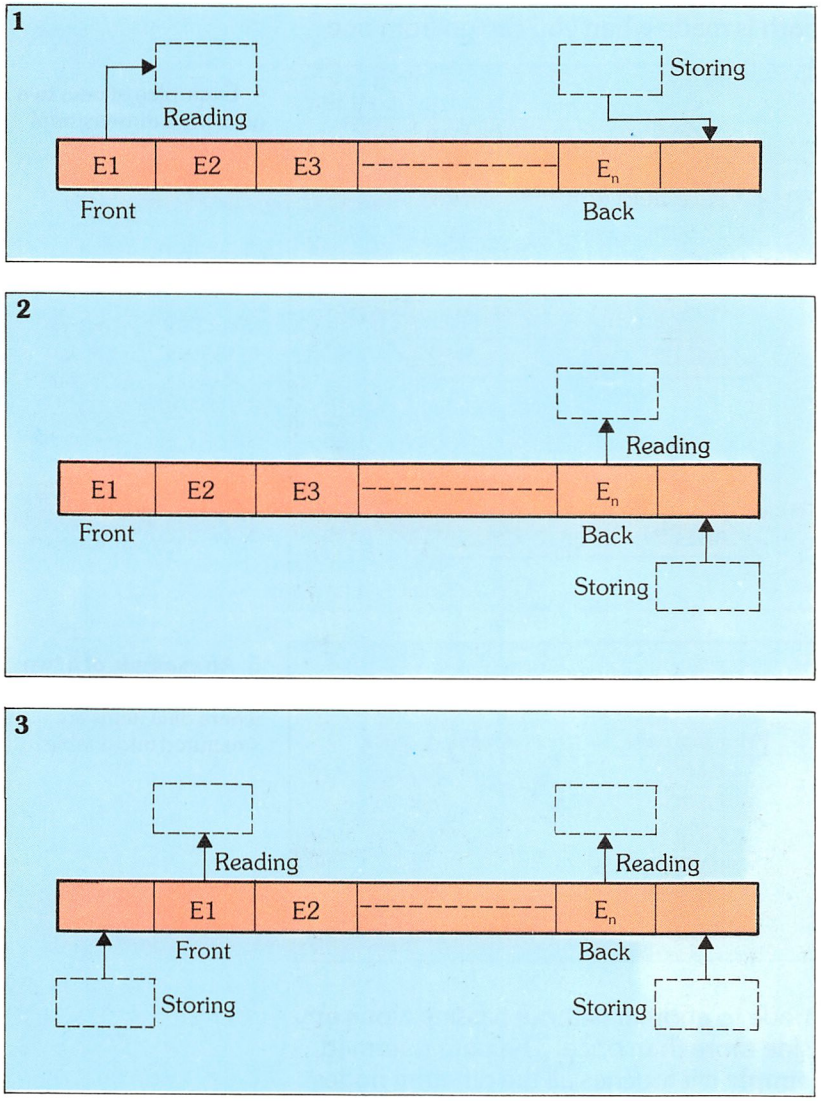
In most cases, the data to be processed does not comprise isolated figures but is linked by some logical relationship. In simple cases, as in the temperature program, the data is all of the same type and belongs to a finished set. In many other cases, the data items are interlinked, but may not all be of the same type.

Data structures are defined as sets of data which are linked by logical relationships. An example could be a personnel record, containing details of a person's date and place of birth, address etc.

The most basic data structure used in computing is the **linear list**, which is also known as a **string**. It is simply a series of data items which follow on from each other. To find a data item contained in a list, you must start from the first data element and sort through it. If the data contained in the list is constant then it is known as a **fixed length list**. If the list can have data added to or taken from it then it is known as a **variable length list**.

The following operations can be performed on lists:
- access to any element;
- insertion of a new element before or after an existing one;
- elimination or deletion of any element.

There are variable length lists which can be operated on in a special way:
1. **Queues**. In queues it is only possible to store new data elements at the end of the list and read them from the front (*figure 1*). The first element that is stored is thus the first to be recalled, so this is often called a **FIFO** (First In/First Out) list.
2. **Stacks**. You can only store and recall data from one end of a stack. The last element to be stored is therefore the first to be read. This is shown in *figure 2*. Stacks are known as **LIFO** (Last In/First Out) lists.
3. **Double queues**. A double queue is represented in *figure 3*. As its name suggests data elements can be stored or read from either end.



1. Queue: an example of a variable length list.

2. Data elements can only be recalled or stored from one end of a stack.

3. A double queue can read or store data elements from either end.

Arrays

In more complex cases, **array** structures are used. An array can have one, two, three or more dimensions, although more than three dimensions are difficult to conceptualise and impossible to draw.

A one dimensional array (also known as a **vector**) is similar to the basic linear list but has the advantage that each element of data is immediately addressable. This is possible because each element has a unique index. For instance, we could have N data items stored in the positions:

X(1), X(2), X(3), ... X(N)

This one dimensional array can be identified by its name X, the indices being the numbers in brackets. If we instructed

the computer to 'READ X(3)', it would be able to directly access this data item. If we were using a linear list to store the data, we would have to sort through it until the third item was reached.

Two dimensional arrays are also known as **tables** or **matrices**. Data held in a two dimensional array can be thought of as being placed in the rows and columns of a table. A matrix of ten rows by ten columns, for example, can hold one hundred items of data. If we were to call this matrix M, then data can be accessed by, for instance, instructing the computer to 'READ M (3,9)'. This would read in the data from the intersection of row 3 column 9 of matrix M. See *figure 4* for examples of one, two and three dimensional arrays.

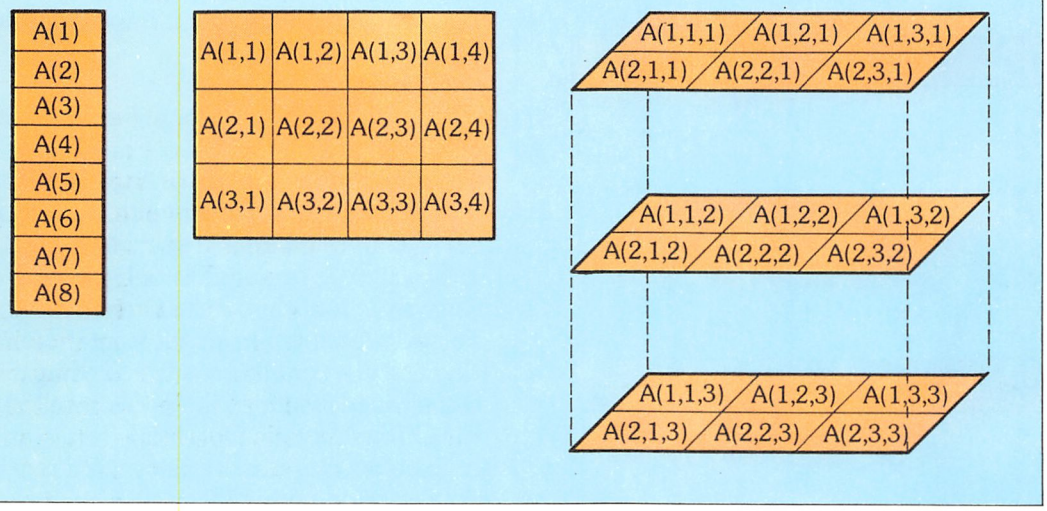
Let's look at the sort of program that could be used to control the stock in a warehouse as an example of a two dimensional array. The list of goods stored in the warehouse could be structured into a table (*figure 5*) which would contain the code numbers of the articles, descriptions of the goods, their price and the number of each held in stock. The article codes would be kept in a preset order for ease of reference.

Where the link between items of data is not simply linear, a more complex data structure is needed. This can be represented by a **chart** (*figure 6*) which consists of a set of linked points called **nodes**. The links between the nodes are called **sides**. A **path** is made when you can go from one node to another without passing along any side more than once. The path is termed **simple** if it touches all the different nodes and returns to the starting node to complete a cycle. If a sense of direction, i.e. ordered priorities, is given to each side, this creates an **oriented chart**, see *figure 7*.

Charts like this, which represent data supported by an operating theory, are widely used to solve practical problems. For example, the use of a chart to find the most economical routes between towns.

Some sets of data can be graphically represented in diagrams which look like upside down trees. For example:

4

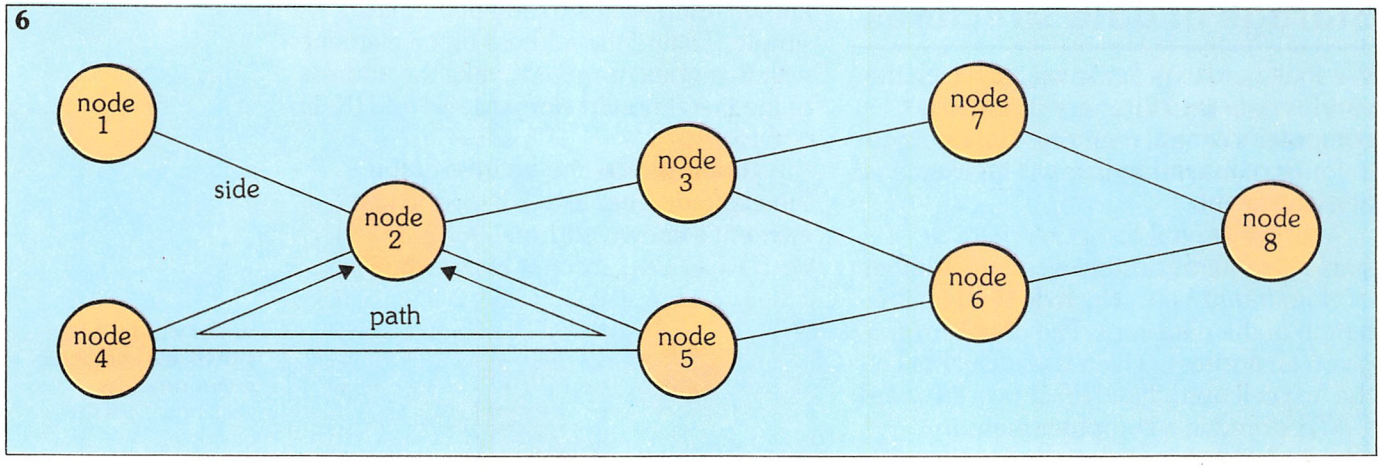


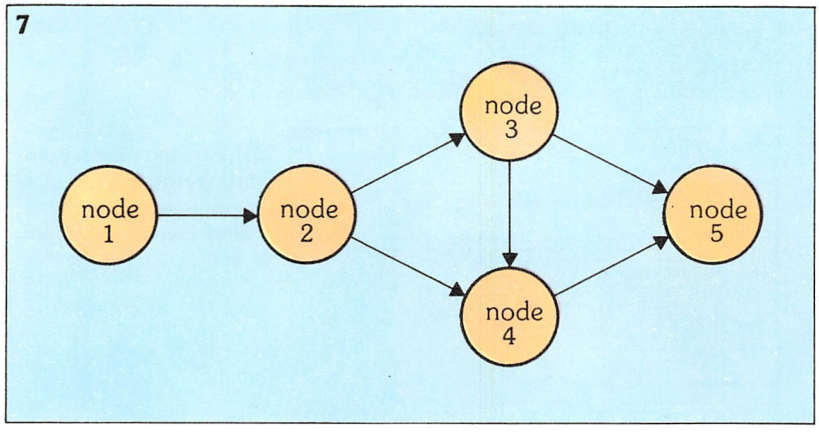
4. Examples of one, two and three dimensional arrays.

5 **Warehouse stock sheet**

Article code	Description	Price per case	Cases in stock
A101	Rich tea biscuits	£3.00	300
A120	Bourbon biscuits	£4.00	450
A200	Ginger nuts	£3.50	7003
B110	Custard creams	£3.75	54
C080	Digestive biscuits	£3.00	3258
F400	Cream crackers	£2.53	1103

5. An example of a two dimensional array where data items are structured into a table.





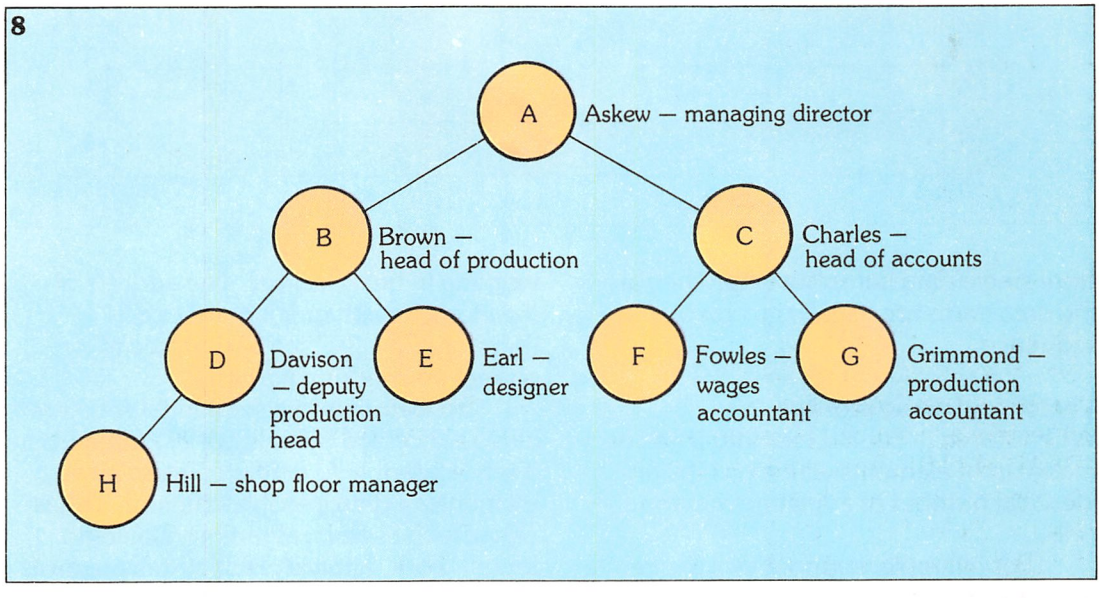
6. A chart consisting of nodes, paths and sides.

7. An oriented chart where a sense of direction is given to each side.

8. An oriented tree.

from the tree you come to a dead end. *Figure 8* shows an oriented tree. The node at the top does not have any sides coming from it – only going into it. This is called the **root node**. The nodes that feed into the root are called **branches**, while the nodes at the bottom of the diagram which do not have any other nodes feeding into them, are called **leaves**. In this case the number of branches per node is two and the tree is known as a binary tree.

As an example of the use of a tree look at *figure 8* again. This shows the

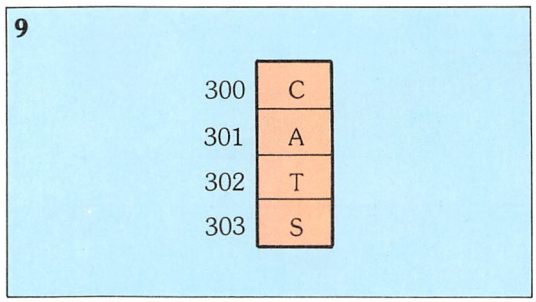


the hierarchy and responsibility lines in a company, family trees or team qualification in a tournament. **Trees** are different from the first type of chart we looked at, as they cannot be used cyclically. That's to say that once a line of data has been traced relationships of responsibility between different people in a company. You can easily trace the superiors to any person in the company by moving up the tree structure, or by tracing down, find out how many departments there are, for example.

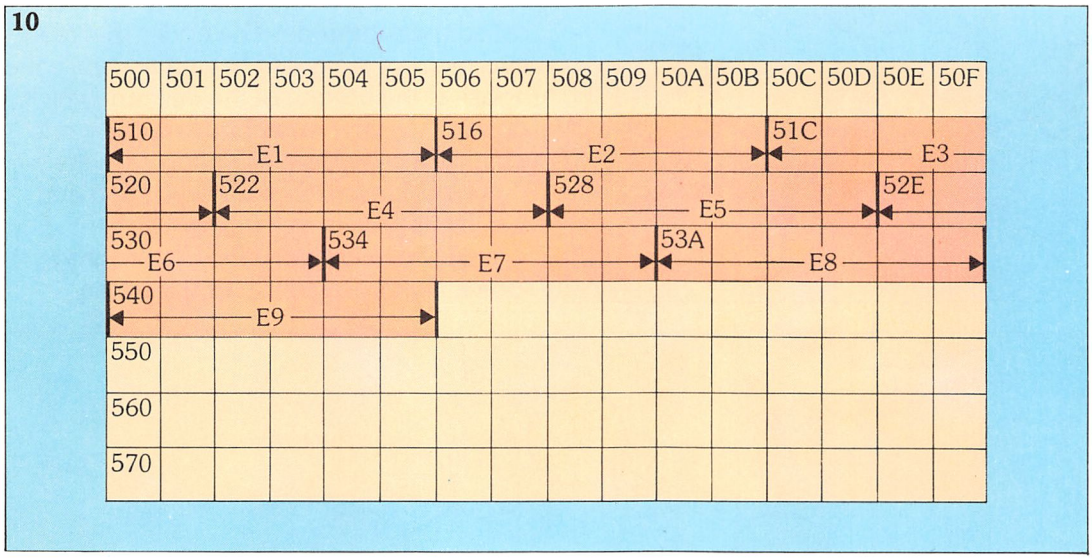
Storage of data structures

We shall now look at the way in which the various data structures are stored in a computer's central memory. Data storage in forms of mass memory will be examined at a later point.

In the central computer memory single data items occupy a certain number of elementary cells, which depend on the length of the data item. The data item is given an address, which is usually that of the first cell used. *Figure 9* shows the word CATS stored in a computer memory whose elementary cell size is one byte. If the computer is programmed in machine language then the code which corresponds to the letters CATS must be written in four bytes. Mass memories are rarely used to store single data items but when they are, the data items are usually grouped in sequence.



9. The word CATS stored in a computer memory whose elementary cell size is one byte.



10. Storing nine 6 byte data items (E1-E9) in a memory where each vector element occupies 6 bytes.

Vector and array storage

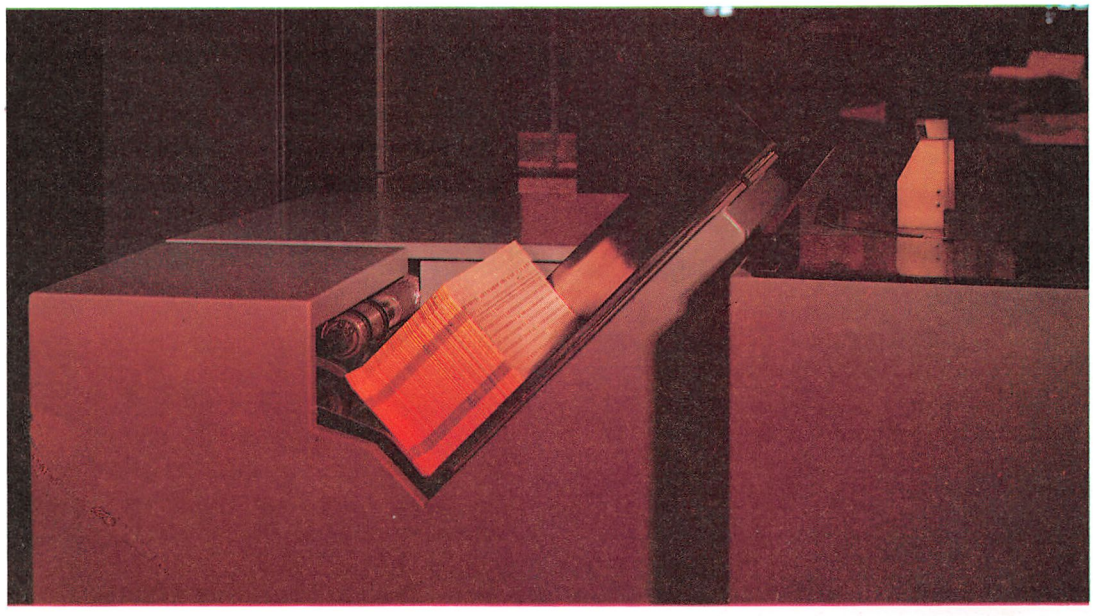
When stored in central memory a vector is made up of elements which occupy an identical number of adjoining memory cells.

For example, suppose that the central memory is organised in bytes and that each vector element occupies 6 bytes. *Figure 10* shows nine 6 byte data items (E1-E9) arranged in store in this way. The small numbers represent the storage addresses and EK the Kth vector element. If the address of the first vector element is known, then access to any of them is simple. To find the address of the element with the index number K, take the address of the first (known) element and add (K-1) $\times$ 6 to it.

For instance, the address of the 7th element is equal to 510_{16} (the first element's known address) + ((7-1)$\times$6), i.e. $510_{16}+24_{16}$ (remember we are working in hexadecimal. The address of the element with index number K is therefore 534_{16} which we can see to be correct by checking *figure 10*.

So elements in a sequential structure can be accessed by a simple algorithm, but it is not possible to delete or insert new elements of data into the structure. This is because the search algorithm would no longer work. Similarly, this system cannot handle vectors whose elements do not have a fixed length, unless the same amount of memory space is allocated to each component of the sequence. Doing this would, however, waste memory space.

A similar procedure can be used to store a two-dimensional array composed



Right: A card reader – this is one method of entering data into the computer. (Photo: Honeywell).

11. The 3 × 4 matrix (b) is stored row by row in memory: **(a)** each unit of data occupies a 2 byte memory position.

11a

290															
300	301	302	303	304	305	306	307	308	309	30A	30B	30C	30D	30E	30F
310	E11	312	E12	314	E13	316	E14	318	E21	31A	E22	31C	E23	31E	E24
320	E31	322	E32	324	E33	326	E34								
330															
340															
350															
360															

11b

$$\text{Rows}\begin{cases}1\\2\\3\end{cases}\begin{pmatrix}\text{E11} & \text{E12} & \text{E13} & \text{E14}\\ \text{E21} & \text{E22} & \text{E23} & \text{E24}\\ \text{E31} & \text{E32} & \text{E33} & \text{E34}\end{pmatrix}$$

$$\underbrace{1 \quad 2 \quad 3 \quad 4}_{\text{Columns}}$$

of fixed length elements. The first thing to decide is whether to insert the data into the array in rows or columns. *Figure 11a* shows how the contents of a 3 × 4 matrix are stored row by row in memory. Each unit of data occupies a two byte memory position and the memory address therefore increases by two for each successive element.

The data elements are numbered Eik: i stands for the matrix row number and k the matrix column number. Therefore the element E23 corresponds to the item of data held in row 2, column 3 of the matrix.

The algorithm for finding the elements in this array is very simple:

Address = 310 + 8 (i-1) + 2(k-1)

or breaking the algorithm down:

$$\text{Address} = \underbrace{310}_{\mathbf{a}} + \underbrace{2 \times 4\,(i-1)}_{\mathbf{b}} + \underbrace{2\,(k-1)}_{\mathbf{c}}$$

Part **a** of the algorithm refers to the address of the first data element. This gives a reference point with which to locate the other elements. Part **b** selects the address of the first element of each row when added to 310. The 2 refers to the number of bytes used by each element, and the 4 to the number of columns. Part **c** selects the data item in the required column. Again the 2 refers to the number of bytes in each element.

Figure 11b shows how these data elements would look in matrix form. As an example, let's find the address of data item

E34 (remember we are working in hexadecimal notation):

Address = 310 + 8(i–1) + 2(k-1)

i = 3 and k = 4

so:

Address = 310 + 8 (3 – 1) + 2 (4 – 1)

= 310 + 10 + 6

Address = 326_{16}

which we can see to be correct by checking *figure 11a*.

These two previous examples concern central memory storage of fixed length data elements. To represent data items of different, known lengths we need to insert an indicator (of fixed length) before each item. This indicator tells us the length of the following data item, which makes accessing relatively easy and saves memory space.

The first data item stored in this manner is easily accessed as the first address is known. For the following items, the length of the element considered + 1 (if 1 is the length of the indicator) must be added to the initial address in a special counter. This enables us to reach the elements' length indicators in sequence.

The disadvantage of this method of data storage is that the elements can only be reached sequentially; it is not possible to access particular elements by the use of an algorithm. Because of this, it is sometimes more convenient to define the length of the storage elements as being equal to the largest data item. The shorter elements can then be filled out with zeros or suitable spaces. It is then possible to use algorithms to directly access data elements.

Sequential memory structures are suitable for storing arrays, tables and linear lists. However they must be composed of constant length elements and cannot be amended once stored.

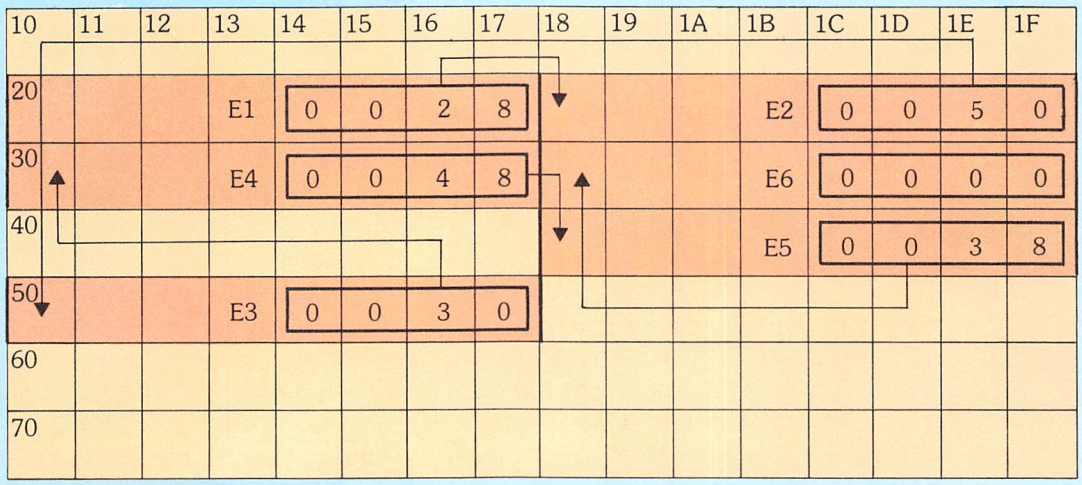
Chained structures

Chained structures provide a more flexible way of storing data. This is a non-sequential storage method, with each element containing a pointer that specifies the memory address of the next data item. *Figure 12* shows an example of chained storage. Each element occupies 4 bytes for the data with another 4 for the pointer. The last element in the chain has its pointer set to zero.

As you can see, the memory space occupied here is double that necessary for data alone. However, it is a more flexible storage structure than those we have looked at previously because it allows data elements to be deleted and inserted. To add a new element to the sequence all you have to do is modify the pointer of the preceding element in the sequence. The preceding pointer is set to the address of the new data element, and the new data element's pointer is set to the figure the preceding element contained.

For an example look at *table 1* which corresponds to *figure 12*. To insert a new

12



12. Chained storage where each data element occupies 4 bytes for the data and 4 bytes for the pointer.

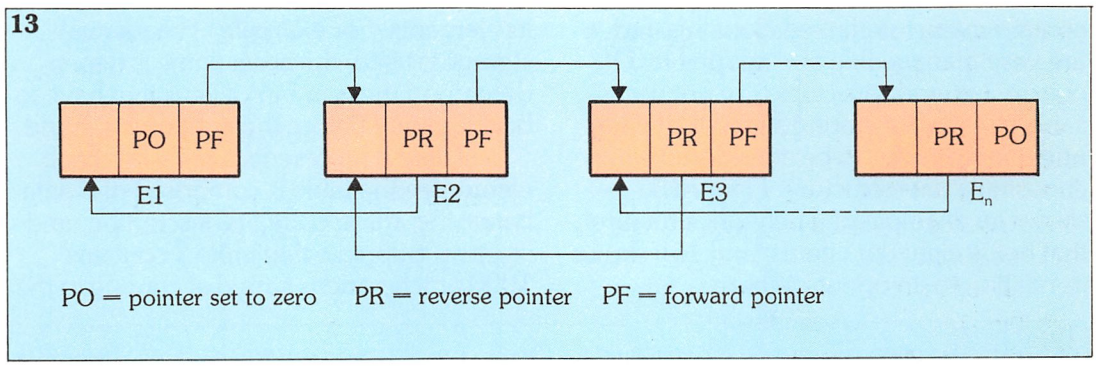
Table 1

Element address	Element	Pointer
20	E1	28
28	E2	50
50	E3	30
30	E4	48
48	E5	38
38	E6	00

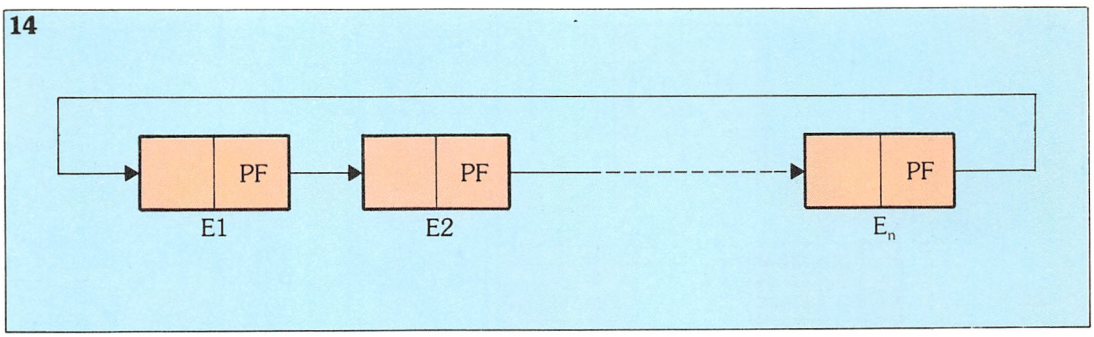
become circular if the last element points to the first and/or vice versa.

To eliminate an element from a chain all you have to do is change a pointer. An element K, for instance, instead of pointing to K + 1 would point to K + 2. Element K + 1 is now eliminated from the chain. In the case of a reversible (bidirectional) chain the backwards pointer of K + 2 will also be changed, in this case to point to element K.

13. A reversible chain.



14. When eliminating an element from a one directional chain, the unoccupied memory space is available for later use.



data item, say after element E3, you need to set the pointer of E3 to 40 (the address of the new element), memorize the new element E31 in the memory location addressed by 40 and set its pointer to 30 (the old pointer from E3).

The data elements that make up this structure can be arranged arbitrarily in memory but when they are first inserted they are stored sequentially. New, out of sequence positions are used in the case of deletions and additions to the chain. Another characteristic of this structure is that it can only be run through sequentially in one direction – that indicated by the pointers. To make a reversible chain, a second pointer must be added to indicate the position of the preceding element. This, however, increases the memory space used. *Figure 13* shows an example of a reversible chain. Remember, chains

Figure 14 shows the effect of eliminating an element from a one directional chain. The now unoccupied memory space is available for later use, if it is reported to the computers' operating system. Always bear in mind the fact that chains do not give direct access to their elements and must always be scanned like sequential storage.

Very complex chained structures can be created. The elements in these chains can have a great number of pointers, each of which with a different meaning. This enables us to store almost any type of data structure.

Network structures

Linear lists and arrays can be stored quite simply using chained structures, while more complex data arrangements such as charts and trees, can be stored in **network structures**. Using network structures the

elements are arranged arbitrarily in memory and each element is composed of:
- format, which includes the data description (length/type) and the description of the pointers (type and number);
- the data item;
- the pointers.

Network structures occupy considerably more memory space than any of the other storage structures outlined so far. However, network structures can contain many interlinked elements and are very manageable and adaptable. Of course, network structures are not necessarily suitable for storing all types of data, and this is a factor to be considered when choosing a data structure. *Figure 15a* shows an example of a network structure that holds eight data items, and *15b* shows its relation to an oriented chart.

Storing tables

One of the most widely used data structures in programming is the table. At the beginning of this chapter we saw that each element in a table is composed of at least two data items: the first representing its access key; the second (or others) representing the data item itself.

The way in which a table is stored determines the time taken to access one of its elements. For example, if sequential storage is used, the access time is dependent upon the number of keys that have to be examined before the right one is found.

Table 2 represents a price list. Each element in the table is composed of 3 data items: the article code; its description; and its price. Suppose that *table 2* contains 1000 articles and is sorted according to the

15. An example of a network structure (a) and the way in which it corresponds to an oriented chart **(b).**

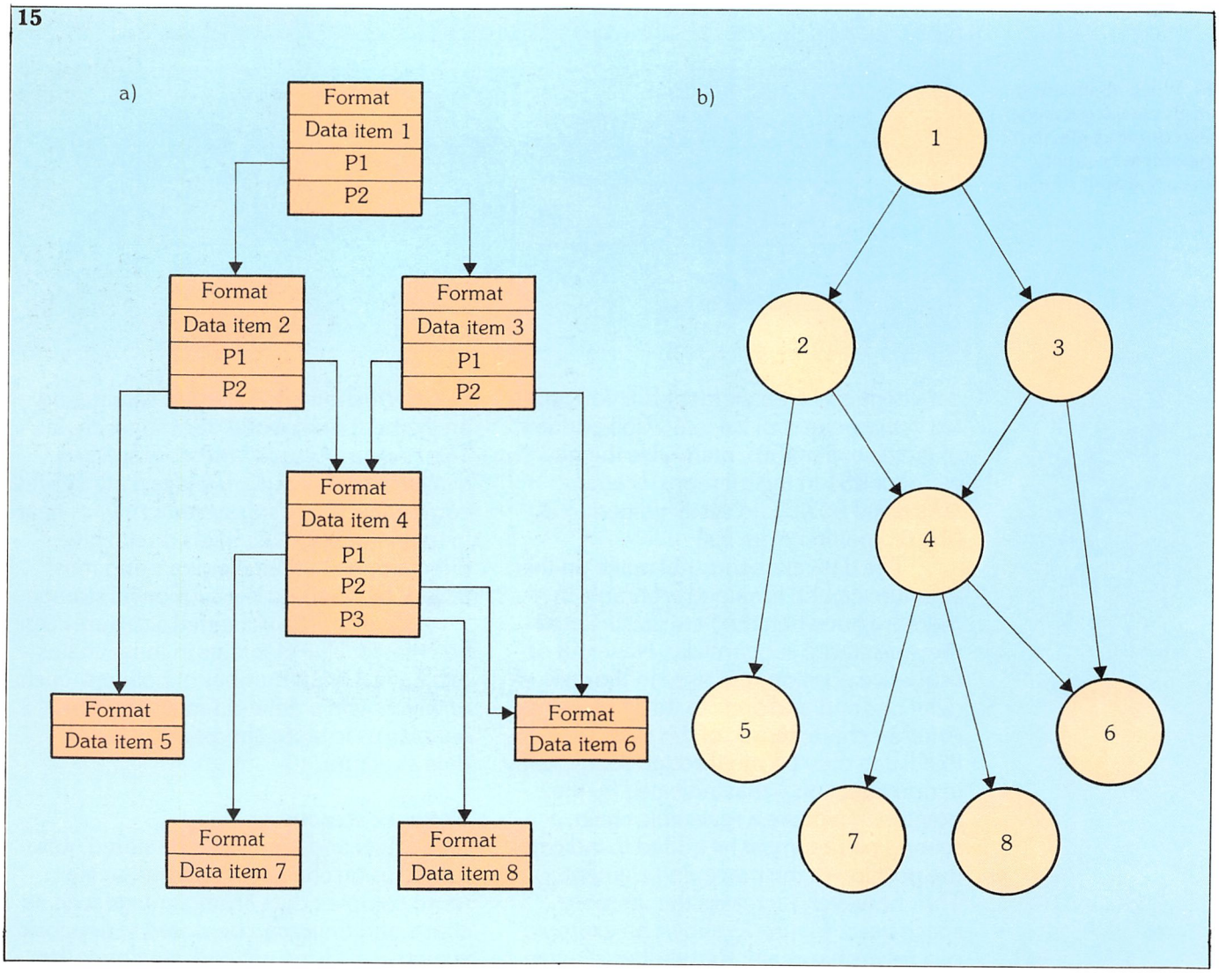


Table 2

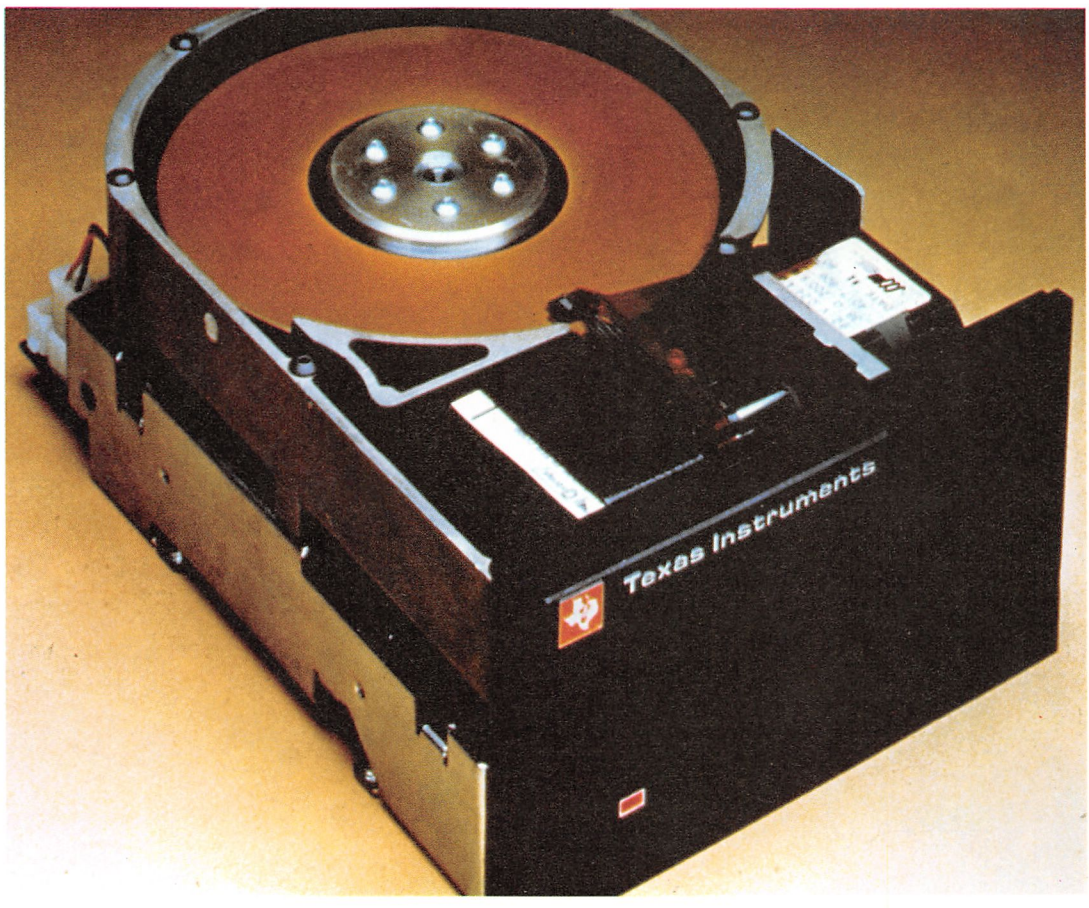
Code	Description	Price
A132	Brogues black	£35.00
A134	Brogues brown	£35.50
B212	Moccasins black	£27.50
B324	Moccasins brown	£28.00

article code (the **search key**). The search key is a code made up of four alpha-numeric characters; the article description is an alphabetic field of 25 characters; and the price is expressed in a numeric field of four characters. Each field has a fixed length.

The three columns of the table are stored using three parallel vectors. This means that the first element of each vector will give the three data elements that make up the first row of the table: code number, description and price. The second element in each vector will give the second row of the table, and so on.

Searching for a code can be done in either of two ways: by a sequential search or by a **dichotomic** or **binary search**. In a sequential search the search key is compared with each of the article code elements in the first vector. As the vectors are parallel, the two other data fields can be read immediately the correct article code is found. If an article code is read that is greater than the search key code, then the article required does not exist in the table. Remember, the article codes are arranged in ascending order of size. This type of search is simple, but slow – in this example, if the required article is last on the list, 1000 comparisons would be made.

The binary search is a faster but more complex searching method. The search begins by comparing the search key to the article code in the centre of the vector. If they match, then the search is over. If they don't match – and the chances are that they won't – then it has to be determined whether the element required is higher or lower than the article code in the centre of



Right: a 5¼" Winchester disk. This is an example of a mass storage device capable of storing 6.38 Mbytes of data.

the vector. This is easily done as the codes are arranged in ascending order. The search key is then compared to the element in the middle of whichever half of the vector the binary search has chosen. Again, if they match the search is over; if they don't match the process is repeated until the code element that corresponds to the key is found. The maximum number of searches that would be made to find an element in a 1000 element vector is 10.

As with the sequential search, once the code element that corresponds to the search key is found, the other elements can be read off from the parallel vectors.

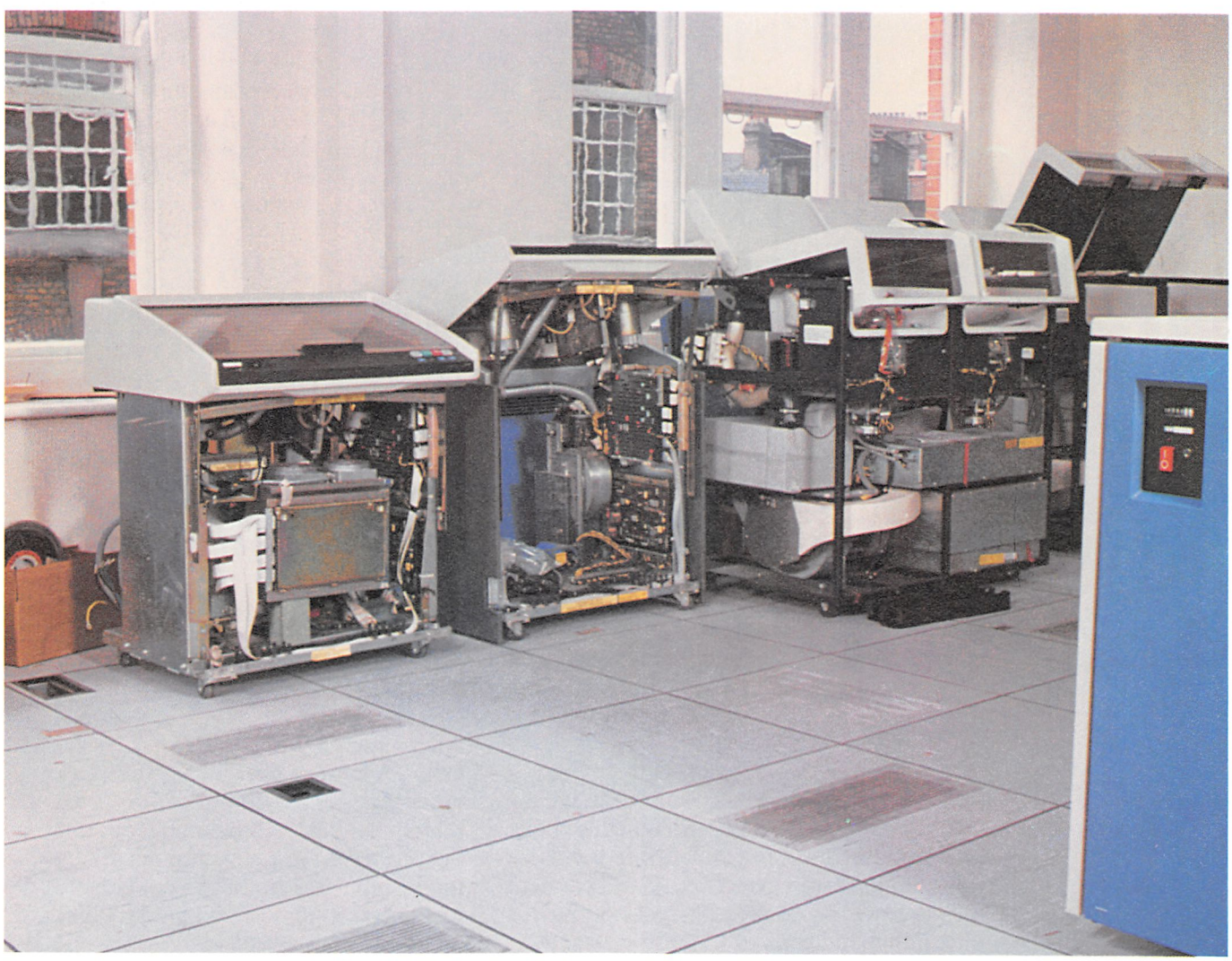
It is important to remember that elements stored in tables must be sorted. If they are not, then all the elements in the table have to be searched through sequentially. Even though the sorting process is a one-off operation, it is still quicker to sort a list and use a binary search than it is to use a sequential search.

An easier way of carrying out a search in a table is when the sorting number of any element in the table can be calculated by means of an algorithm. The same algorithm must obviously be applied when the table is stored.

Occasionally the search algorithm will generate the same result for two or more different keys. This difficulty is overcome by storing these data elements in a separate memory zone, from which they will be accessed by a second routine. Luckily this situation does not arise often enough to present a major problem in most data storage situations.

Another disadvantage of this method of access is wastage of memory space, however this is compensated for by the very fast access time.

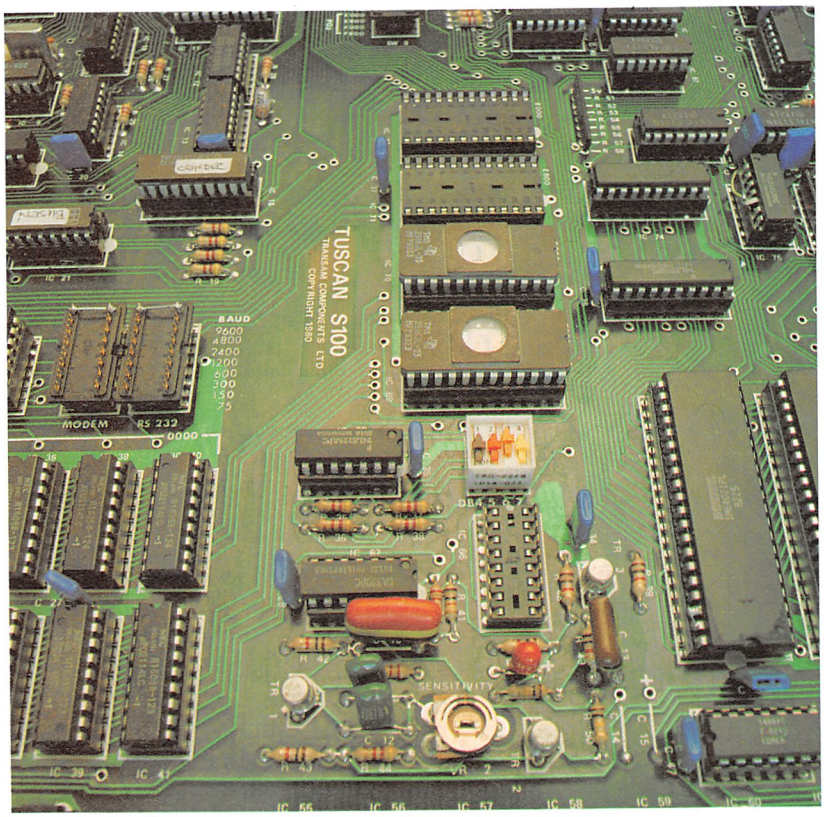
Below: a mainframe computer system prior to installation, showing disk drives on the right of the picture.



Ace Photo Agency/Alan Brockway

Communication between program and stored data

Data can either be placed in a computer memory manually or automatically. To insert data manually, a panel of switches is used to set each individual memory location to the required value. This is a long and laborious process only necessary in very early models.



Science Photo Library/Martin Dohrn

Above: Transam Tuscan 100 computer processor board showing EPROM chips (2 blocks with white circles), video display section (colourful rectangle at front), and memory chips (rectangles on left).

Modern machines now perform this task automatically. The process of data storage is carried out by the specific high level symbolic instruction language used by the computer. Most personal computers use the language BASIC, and the user communicates with the memory by using **symbolic variables**. A symbolic variable is a name given to an item or items of data. These can represent whole numbers, decimal numbers, strings of characters, whole or decimal numbers organised in mono or multidimensional arrays, or strings of characters organised in mono and multidimensional arrays.

The algorithms used for finding specific addresses containing data, are carried out by the computer's operating system on command of the user's program. This means that the user does not have to program the actual algorithms but merely instructs the computer to operate them, for example, by using the instructions 'READ X' – where 'X' is the symbolic variable required.

Remember that a computer system does a considerable amount of work on data, simply in terms of handling it. In fact before data items are stored they are received from a remote input – usually in a form not suitable for storage. Let's look at what happens to data input from a card reader.

In an earlier chapter we saw that a punched card is read a column at a time. The sequence of bits represented in each column – represented in Hollerith code – is held in a memory buffer (remember, each column of bits represents one alphanumeric character). When the sequence of bits is read from this buffer it has to be translated into the code used by the computer, e.g. ASCII, and stored as a byte of information. If the item of data was made up of ten columns of characters on the card, it would be stored as ten consecutive bytes and will be known by the symbolic name assigned to it. All of these operations would be handled by the computer's operating system without the user knowing anything about it.

Before ending this chapter it must be emphasised that there are two problems concerning the organisation of data to be processed. These are the definitions of the logical and the physical structures of the data.

The logical definition of the data struture is the way in which the data is seen by the user, in other words the form the data takes – a string, a one dimensional array, a multidimensional array etc.

The physical definition of the data concerns the way the data items are actually stored in the computer memory, and the problems of space allocation and accessing that go with it.

We shall be looking at the way the logical organisation of data affects the physical working of memory storage in greater detail in a later chapter.

Glossary

algorithm	a set of rules or steps that are followed to perform a certain function, i.e. accessing a particular variable from a list
array	storage structure of one or more dimensions. Each data item is uniquely addressable by means of an identifier
binary or dichotomic search	also known as hashing. A search method that works by dividing the search area into half, then half again and so on until the address required is found. The search algorithm decides which half of the list to look at by comparing the address of the middle item of each portion of the storage area and looking higher or lower, depending on the size of the found address compared to the desired address
chart	a way of representing data – comprised of linked nodes. If the links have a direction then it is known as an oriented chart. If you can travel 'around' the chart returning to the starting position, the chart is cyclical
chained structure	a flexible linked data storage structure, in which each element of data contains a pointer that specifies the memory address of the next data item
data structure	a set of data that is organised for accessing in a specific way
double queue	a variable length list in which data can be stored or read from either end
element or field	the most basic collection of data – a single data item
field length	the number of characters that a data element is made up of
fixed length list	a basic data structure, consisting of a fixed number of data items
linear list	the most basic data structure used in computing – also known as a string. It is simply a series of data items
matrix	an arrangement of data into rows, columns and planes. Each data item can be addressed by its co-ordinates
pointer	an address of a storage location that is usually stored with the preceding data item. It allows data to be randomly stored, but accessed in sequence
queue	a variable length list in which data is stored at one end and read from the other. Also known as a FIFO (first in – first out) list
stack	variable length list in which data can only be stored or read at one end. Also known as a LIFO (last in – first out) list
table	a two dimensional array
tree	a non-cyclical data structure which can be represented in a form like a family tree
vector	a storage structure made up of elements which occupy an identical number of adjoining memory cells